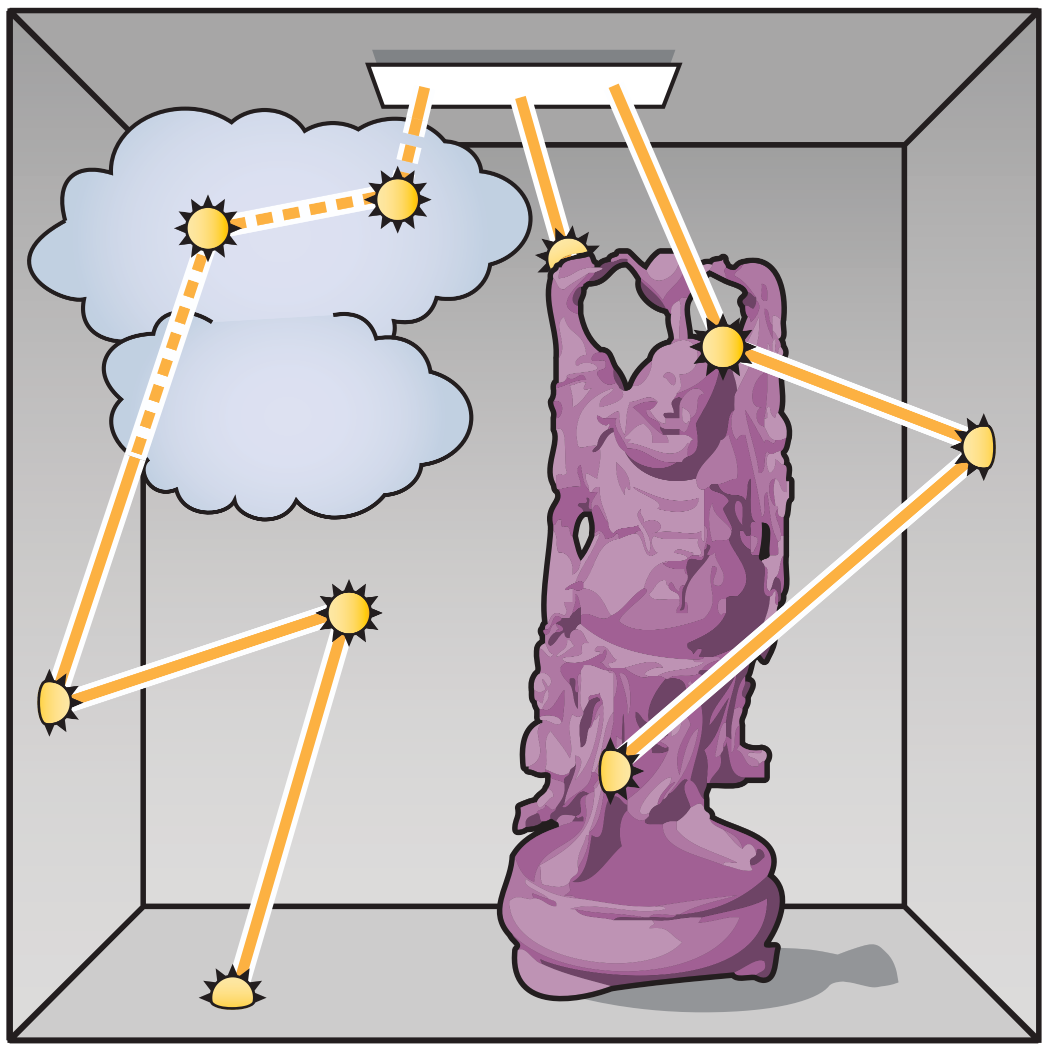
Overview

… follows the structure of the STAR

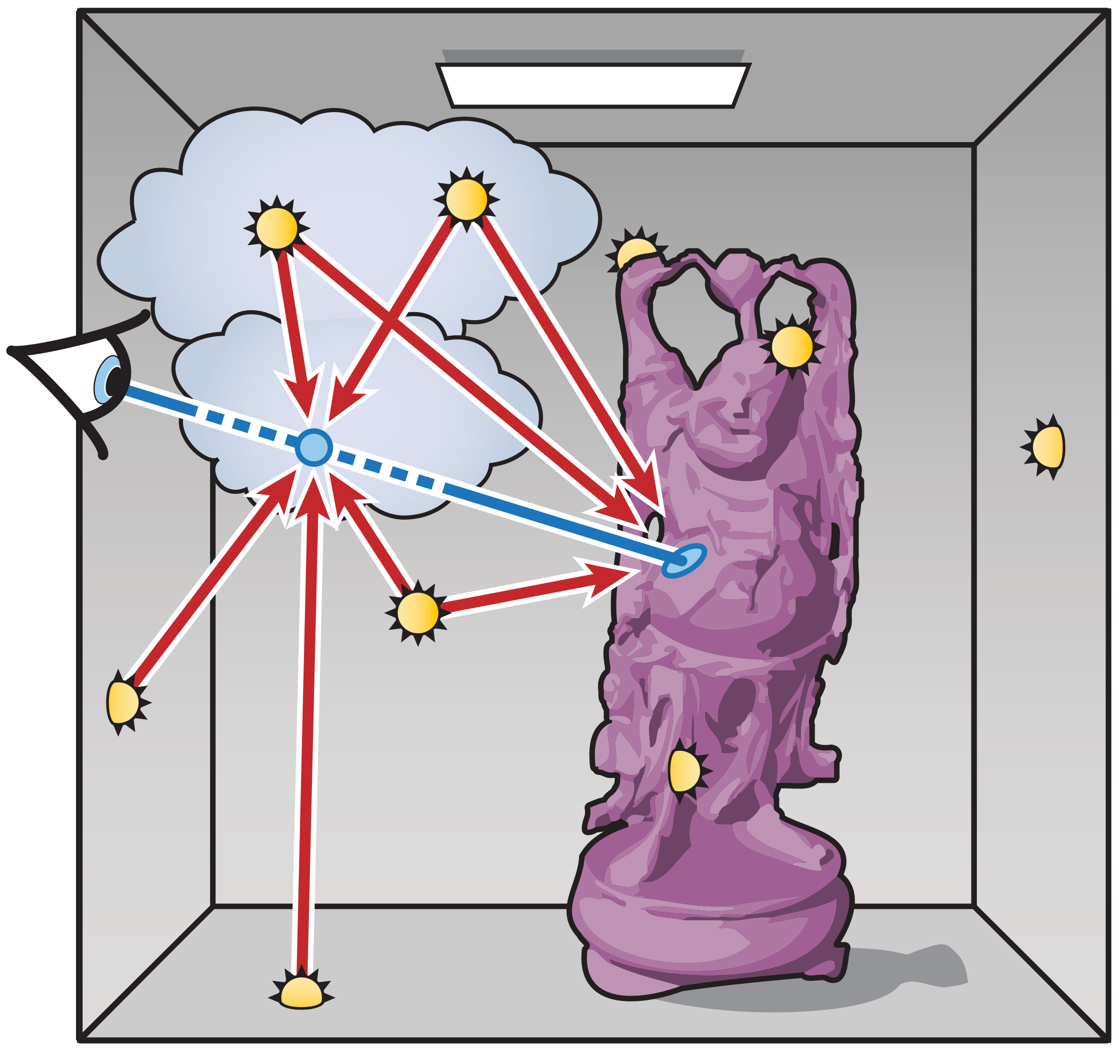
- Introduction & Welcome (Carsten)
- Many-Light Rendering Concepts (Jan)
  - Basic Idea
  - Improved Virtual Lights Generation
  - Lighting with Virtual Lights
- Really Many Lights: Scalability (Carsten)
- Interactive and Real-Time Rendering (Carsten)
- Conclusions, Outlook, Q&A (Jan & Carsten)

Many-light Methods

Two major passes



Generation of VPLs

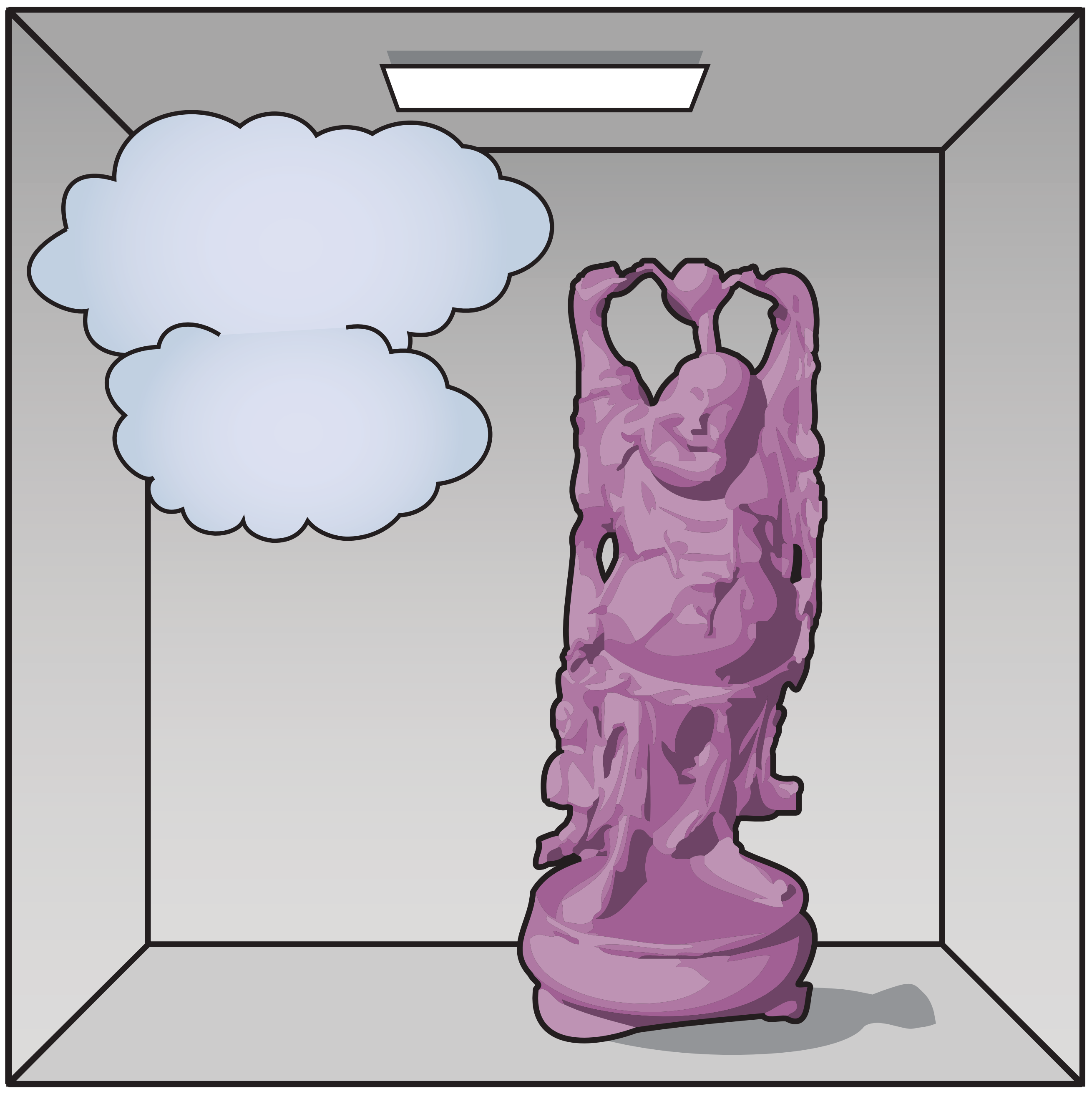


Lighting with VPLs

Many-light methods consist of two passes: generation of VPLs and lighting with VPLs.

Many-light Methods

Pass 1: Generation of VPLs

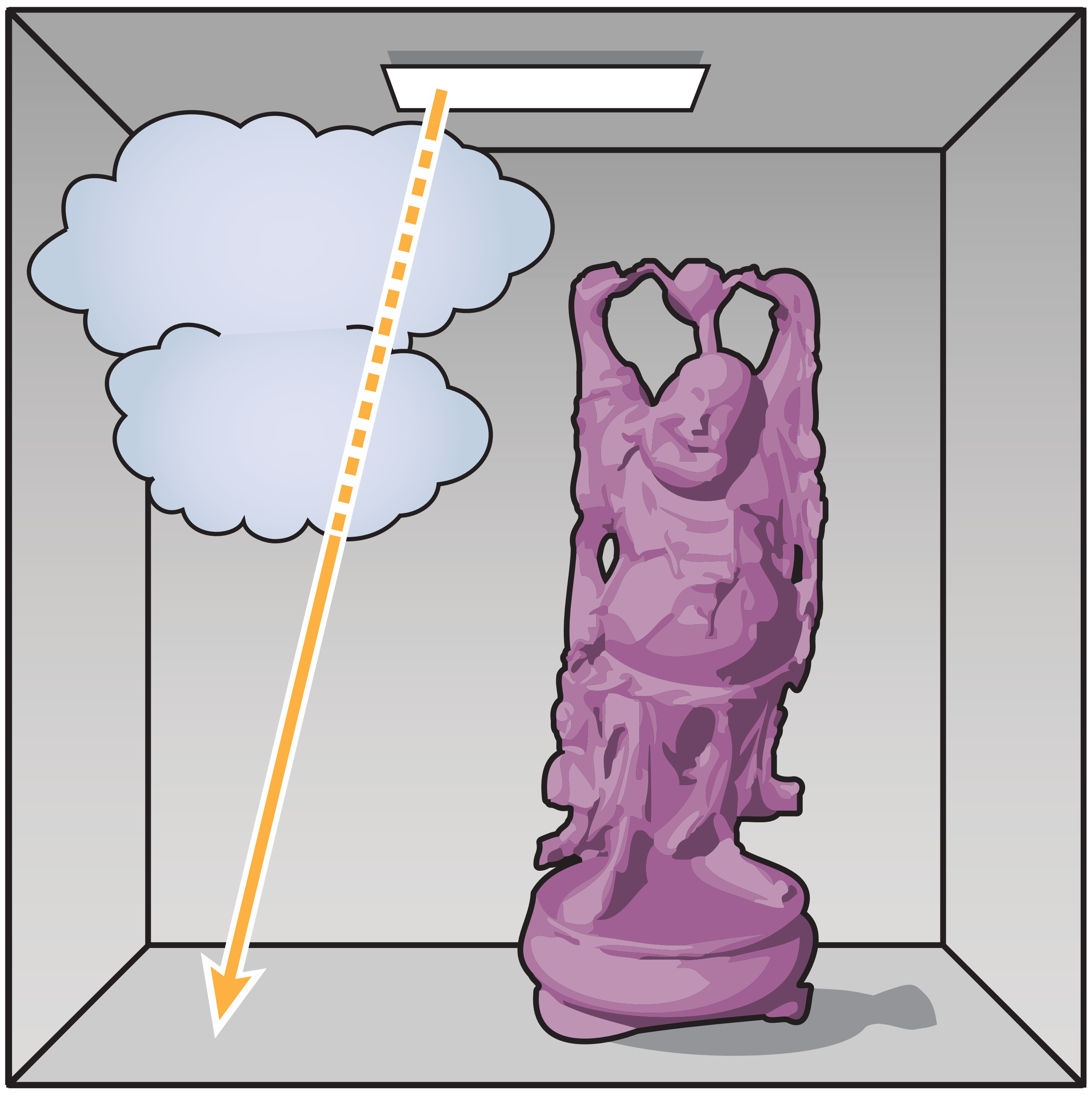


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

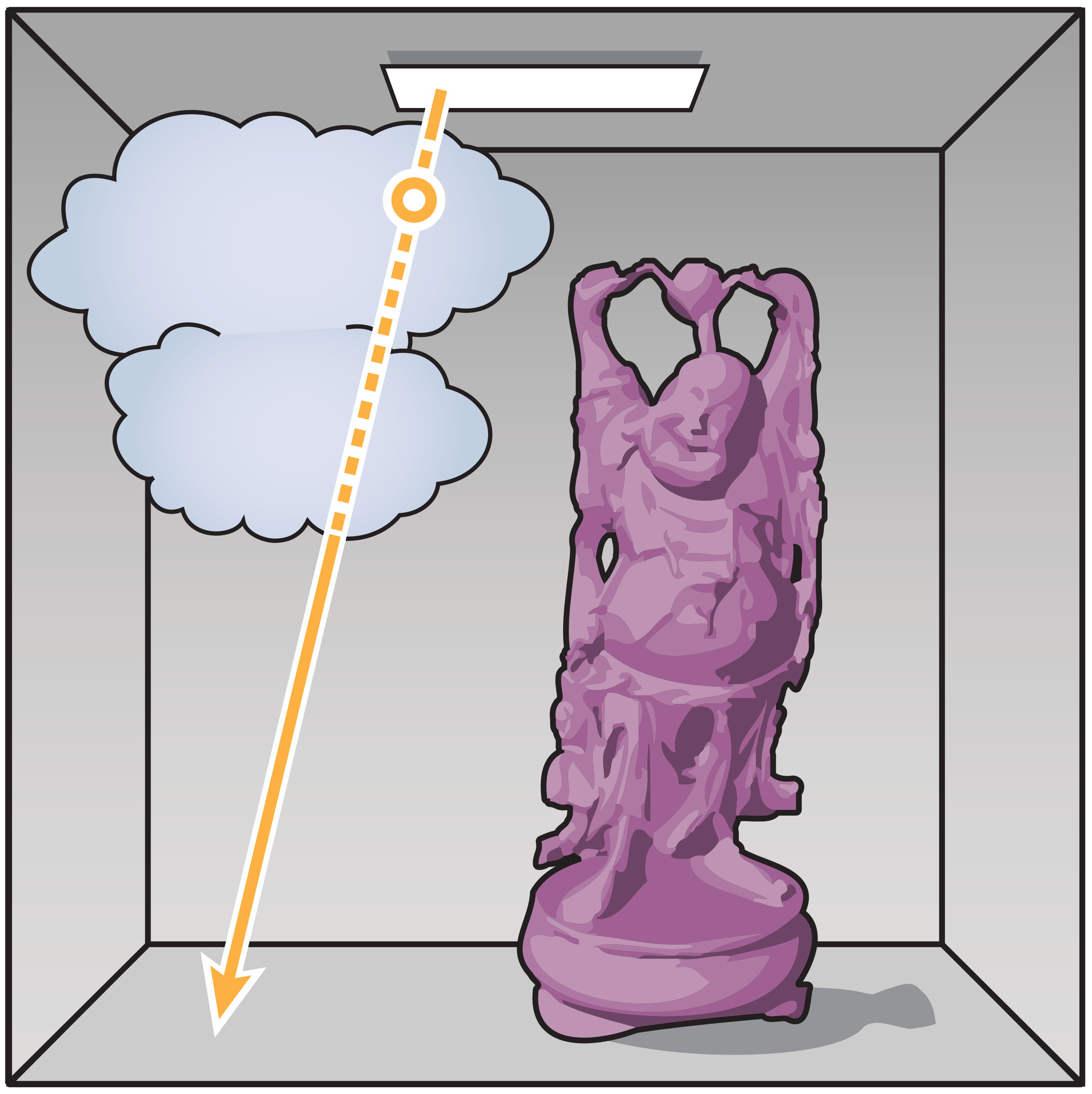


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

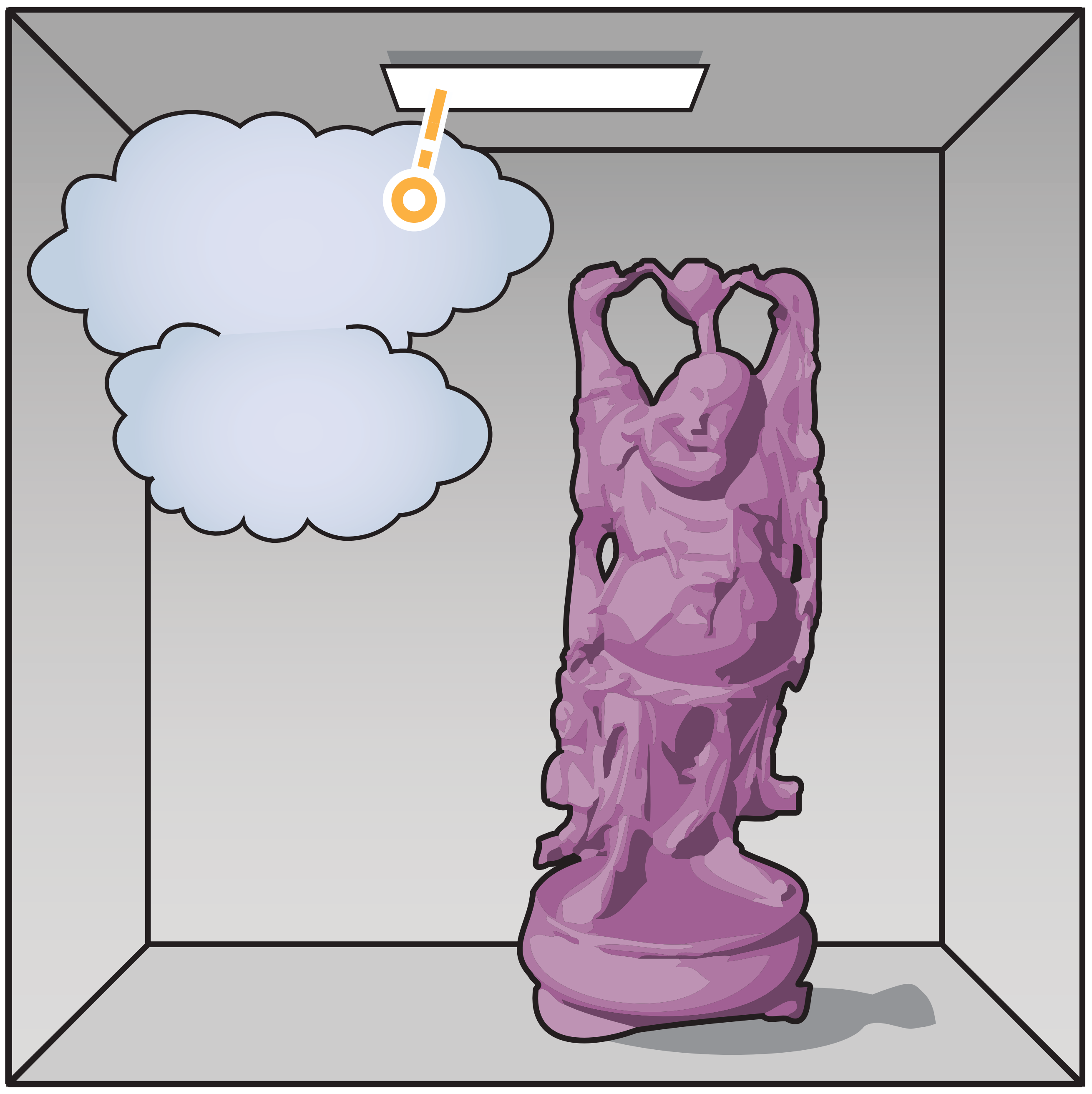


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

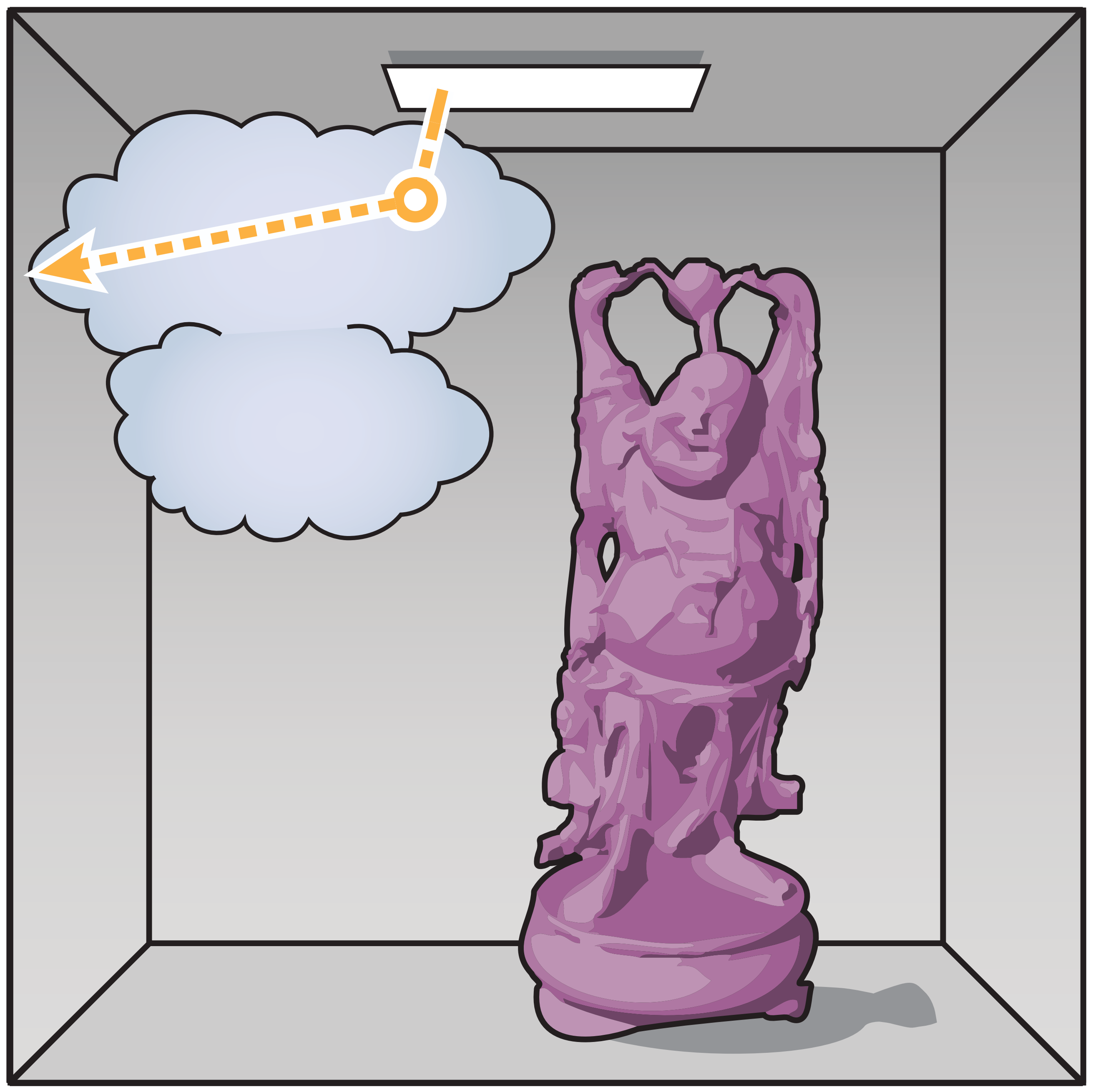


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

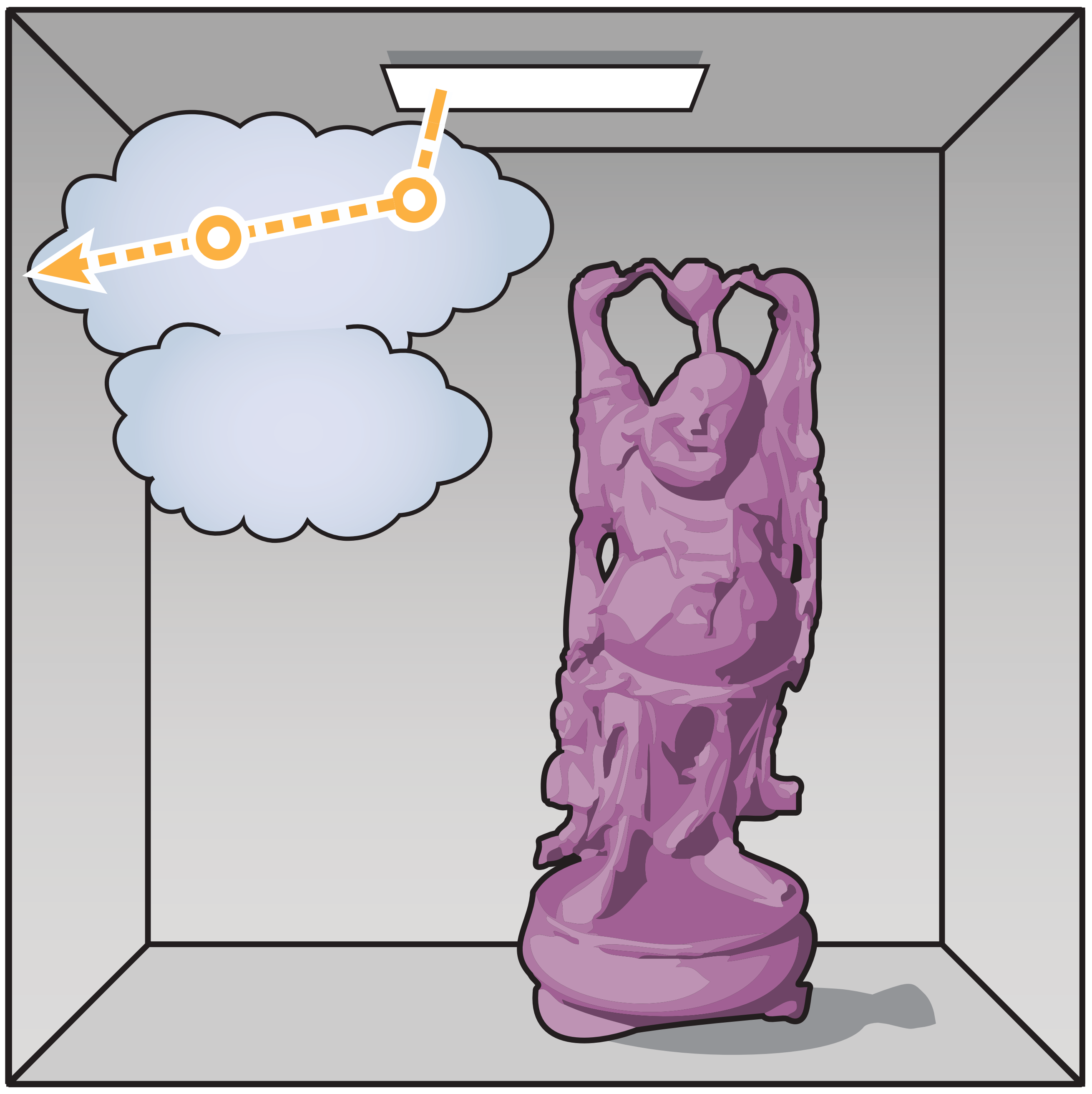


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

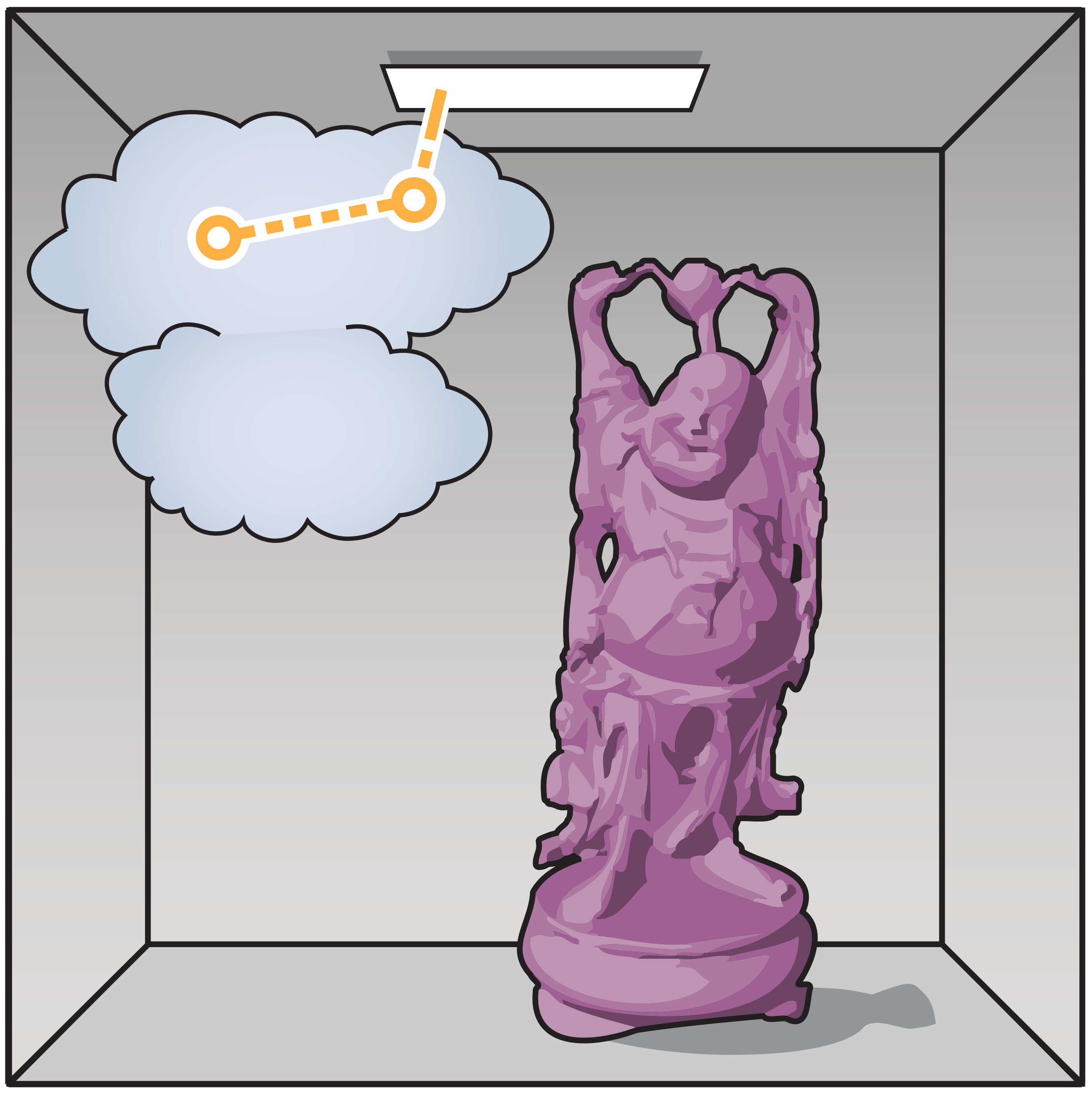


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

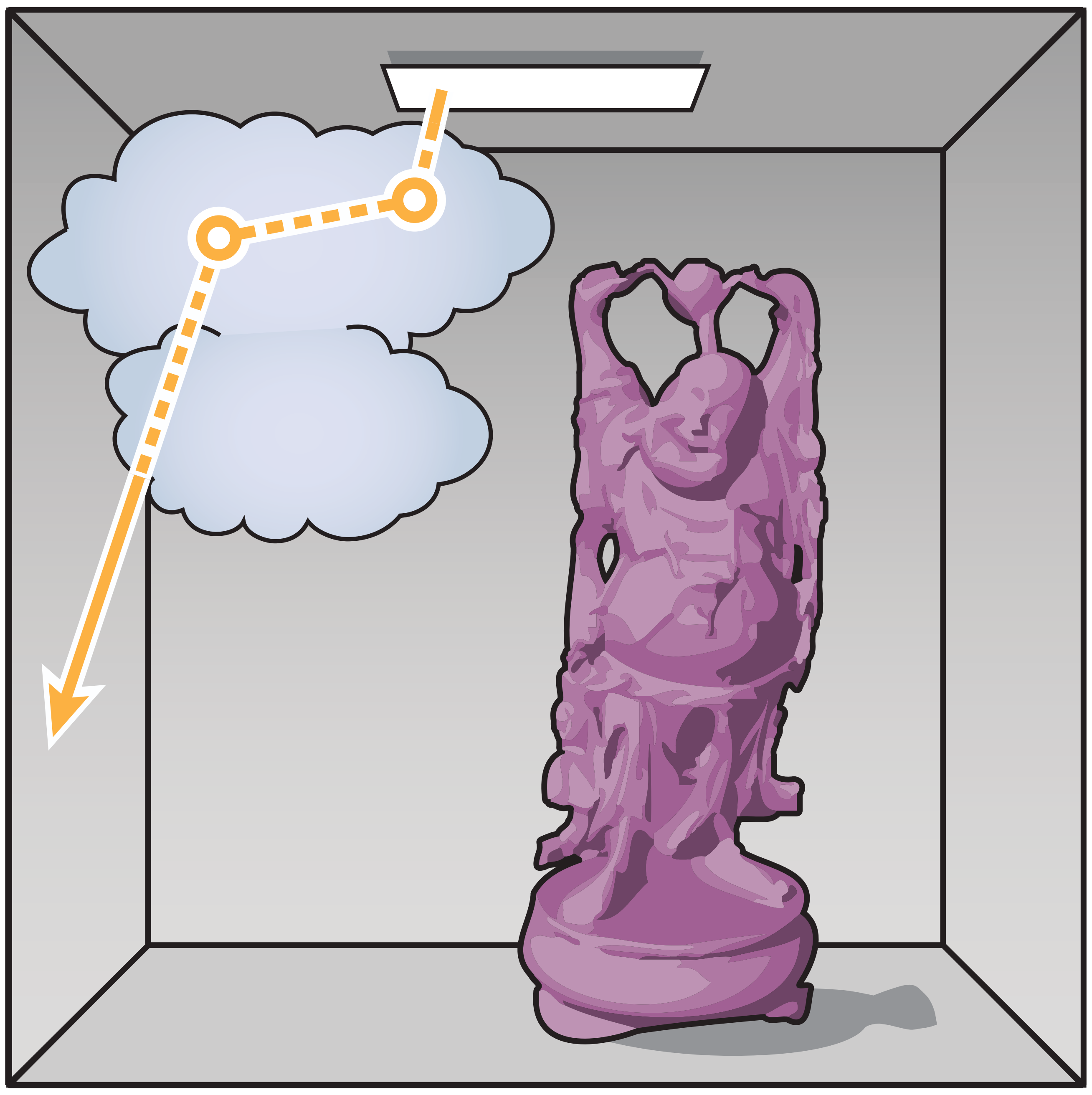


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

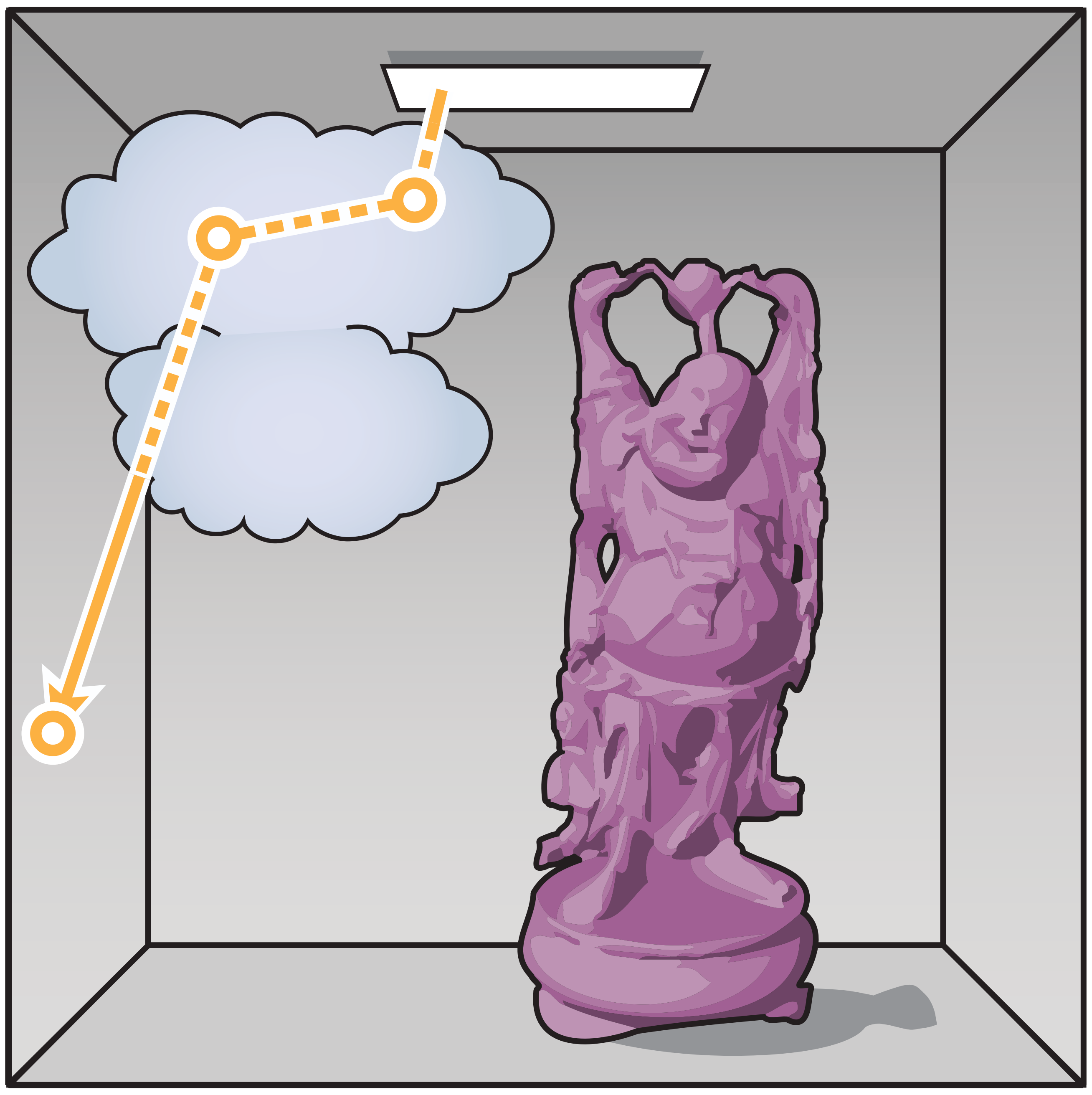


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

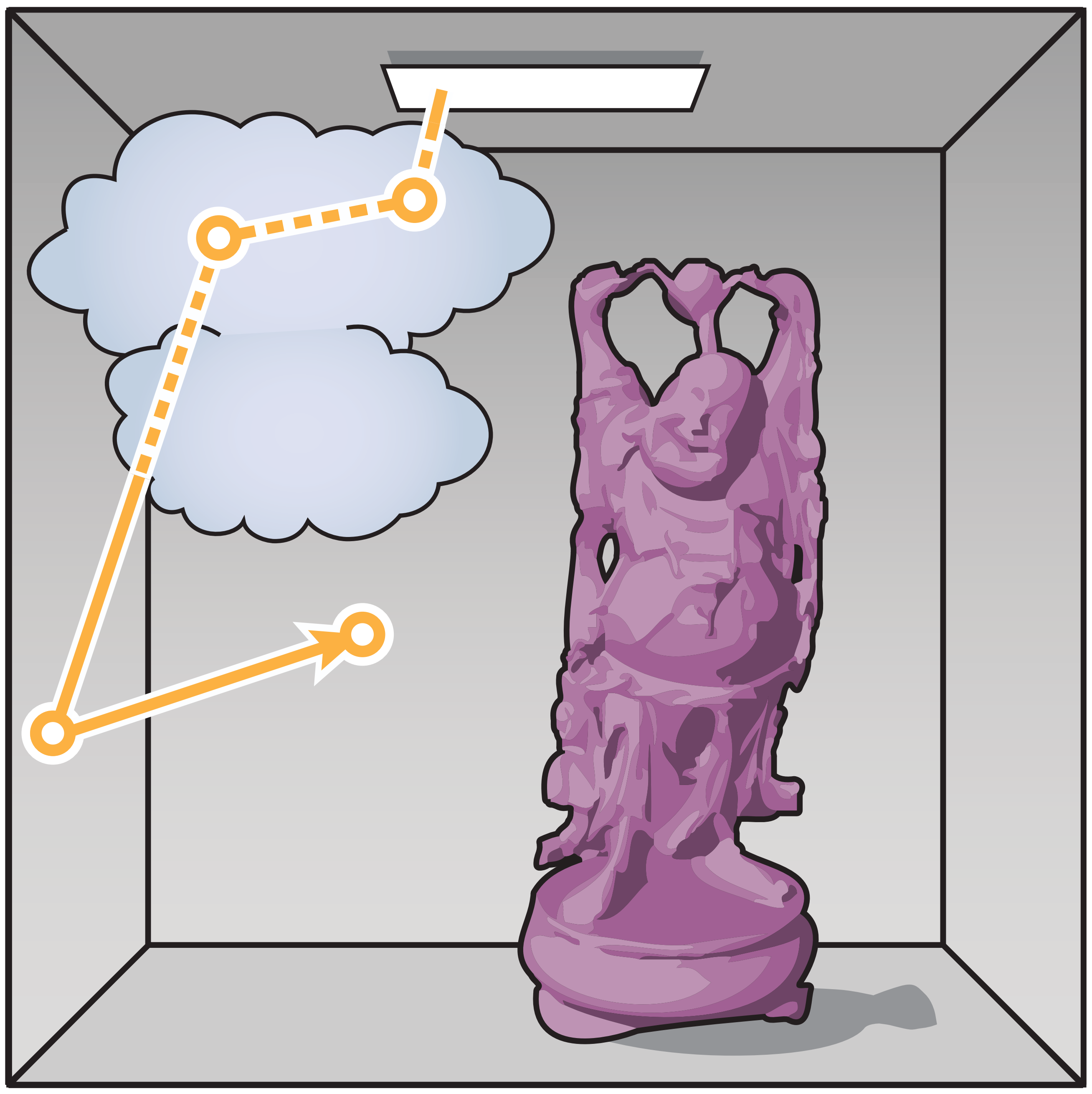


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

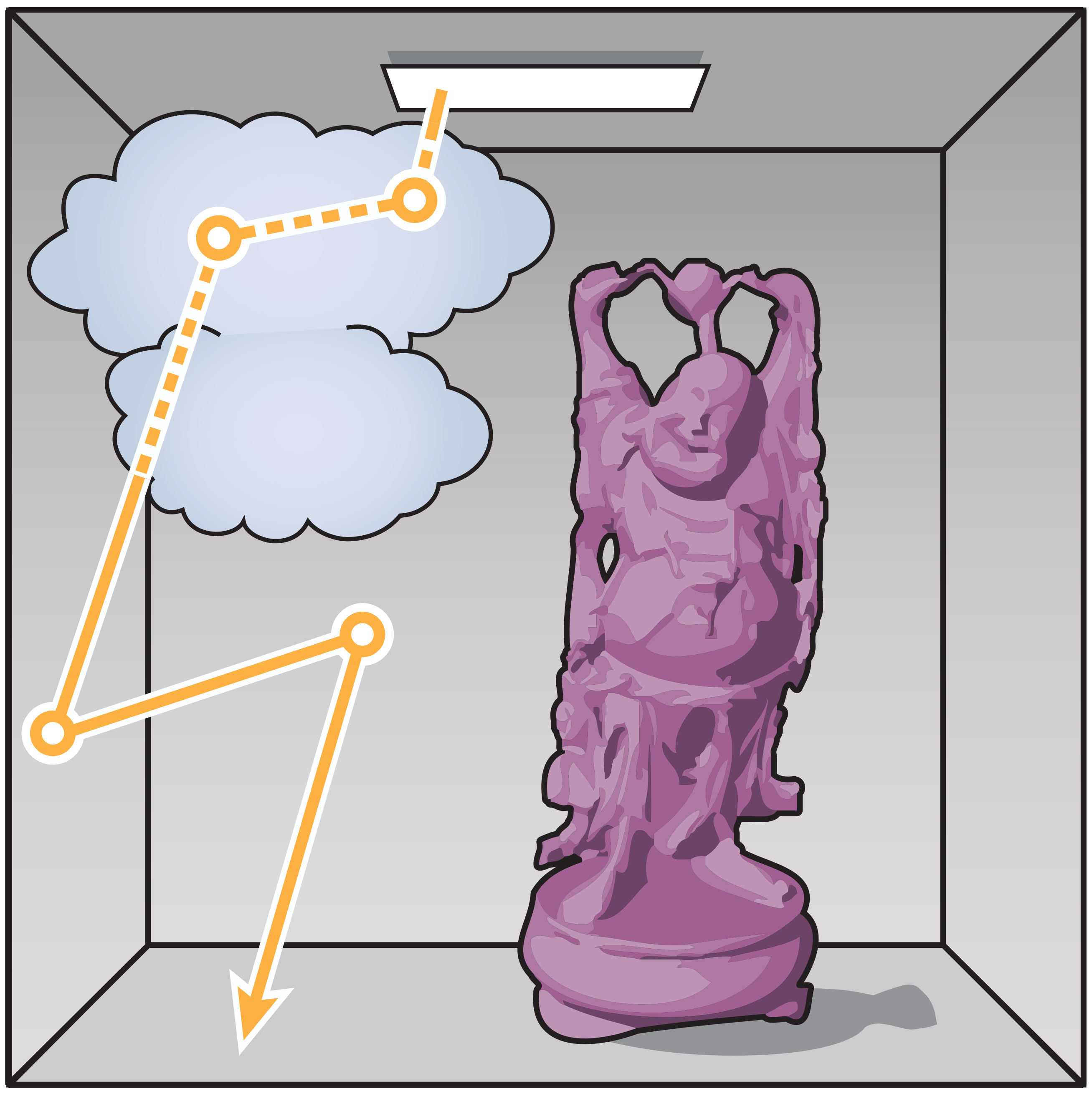


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

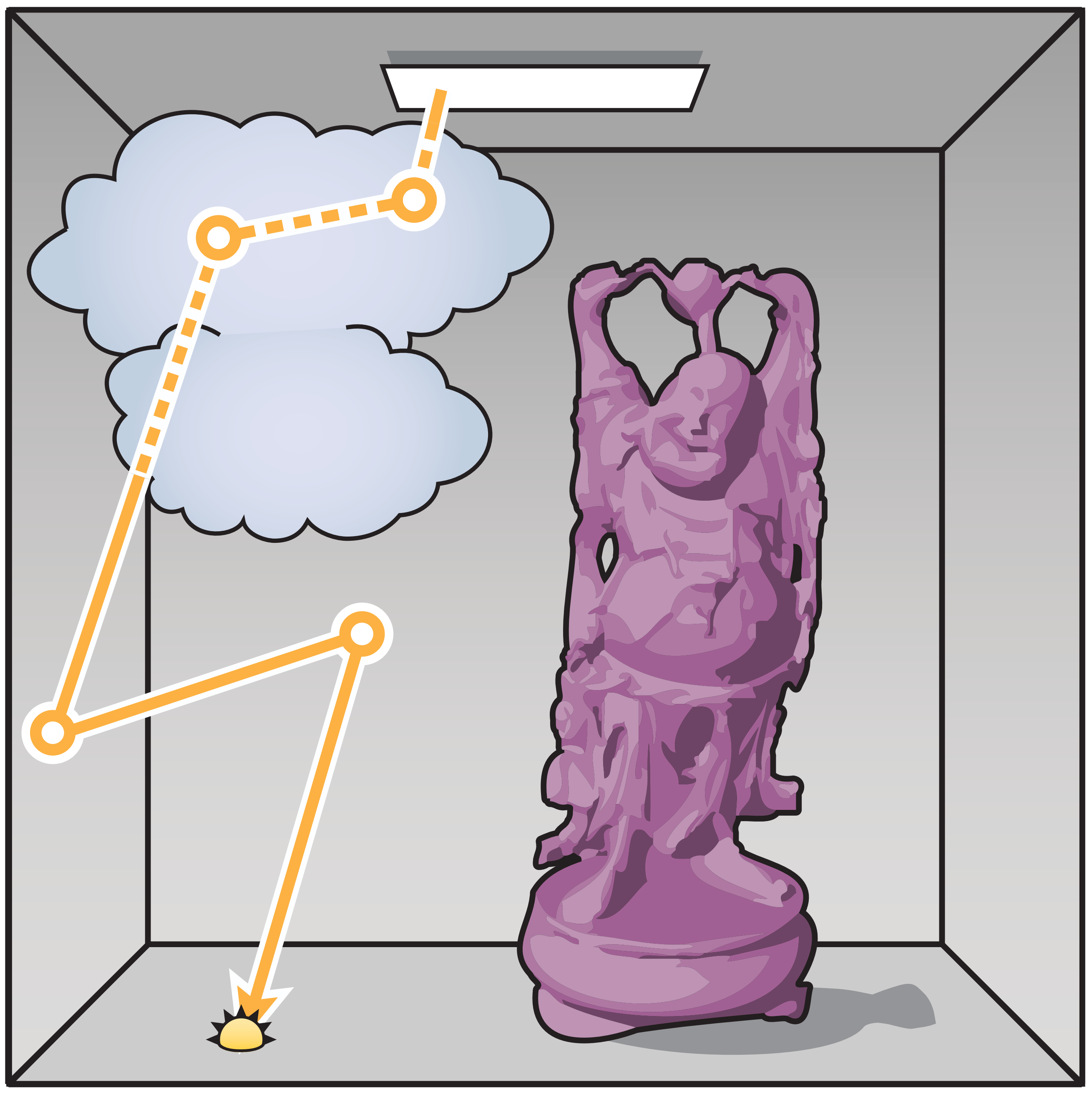


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

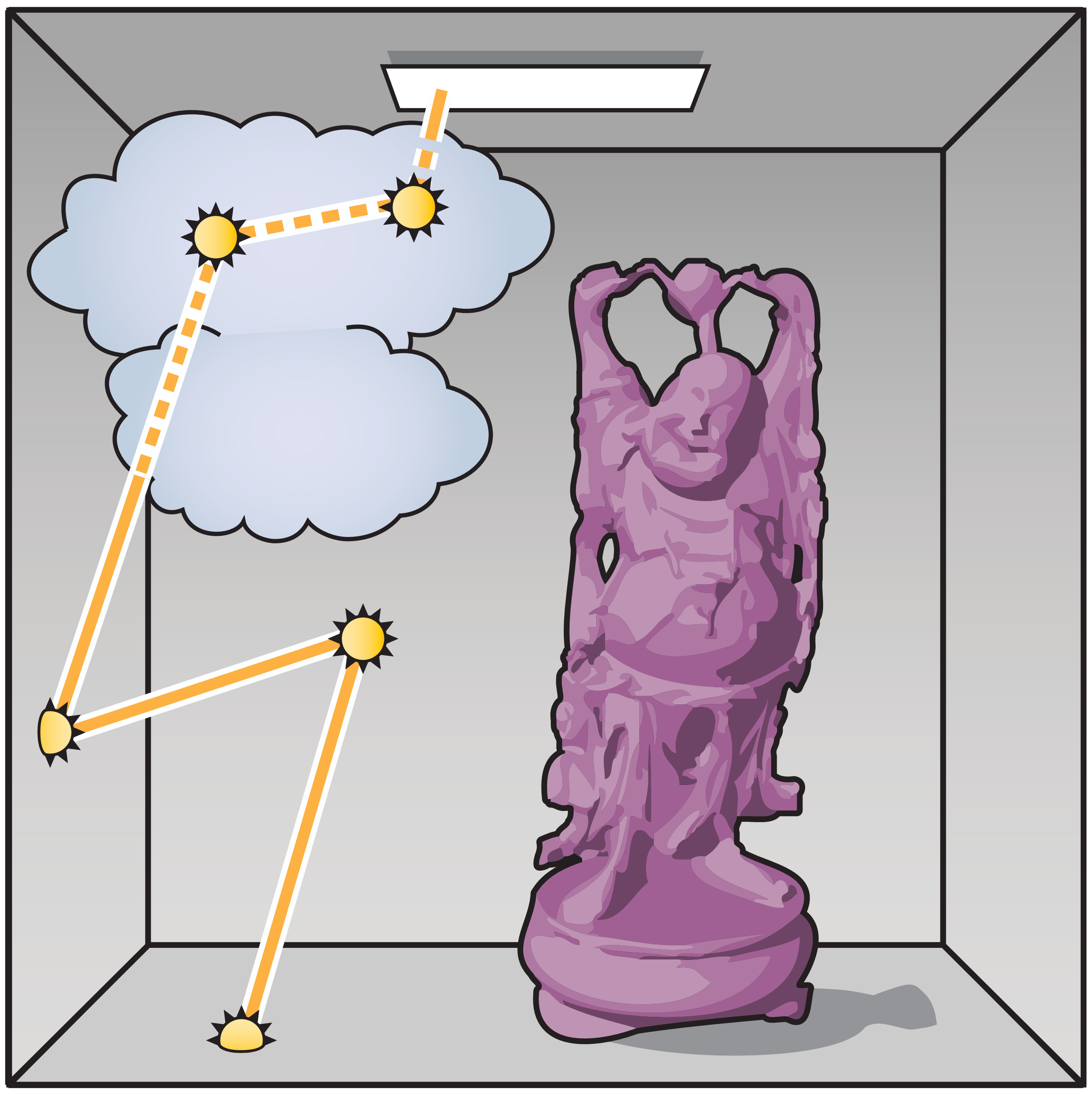


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

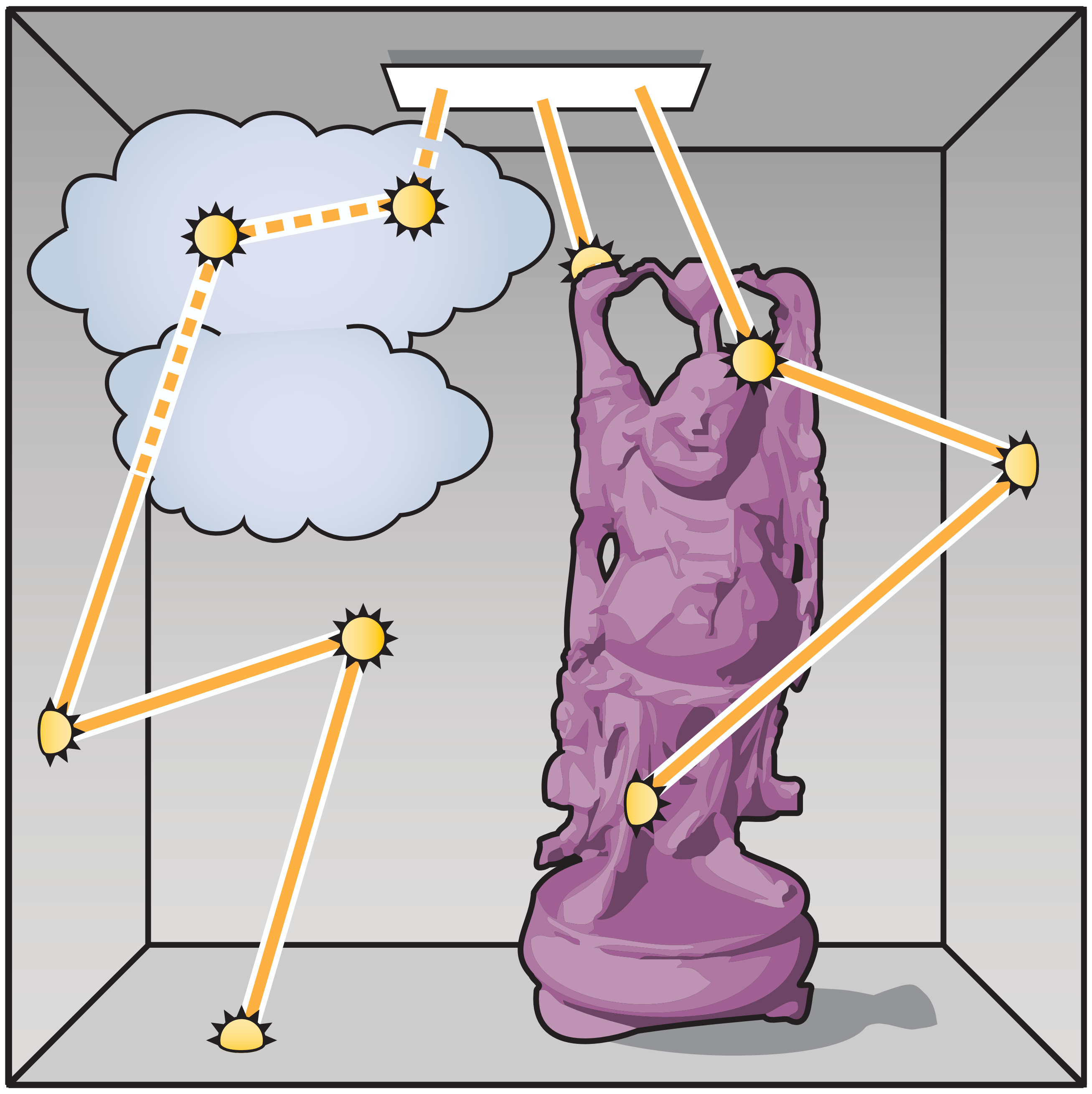


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

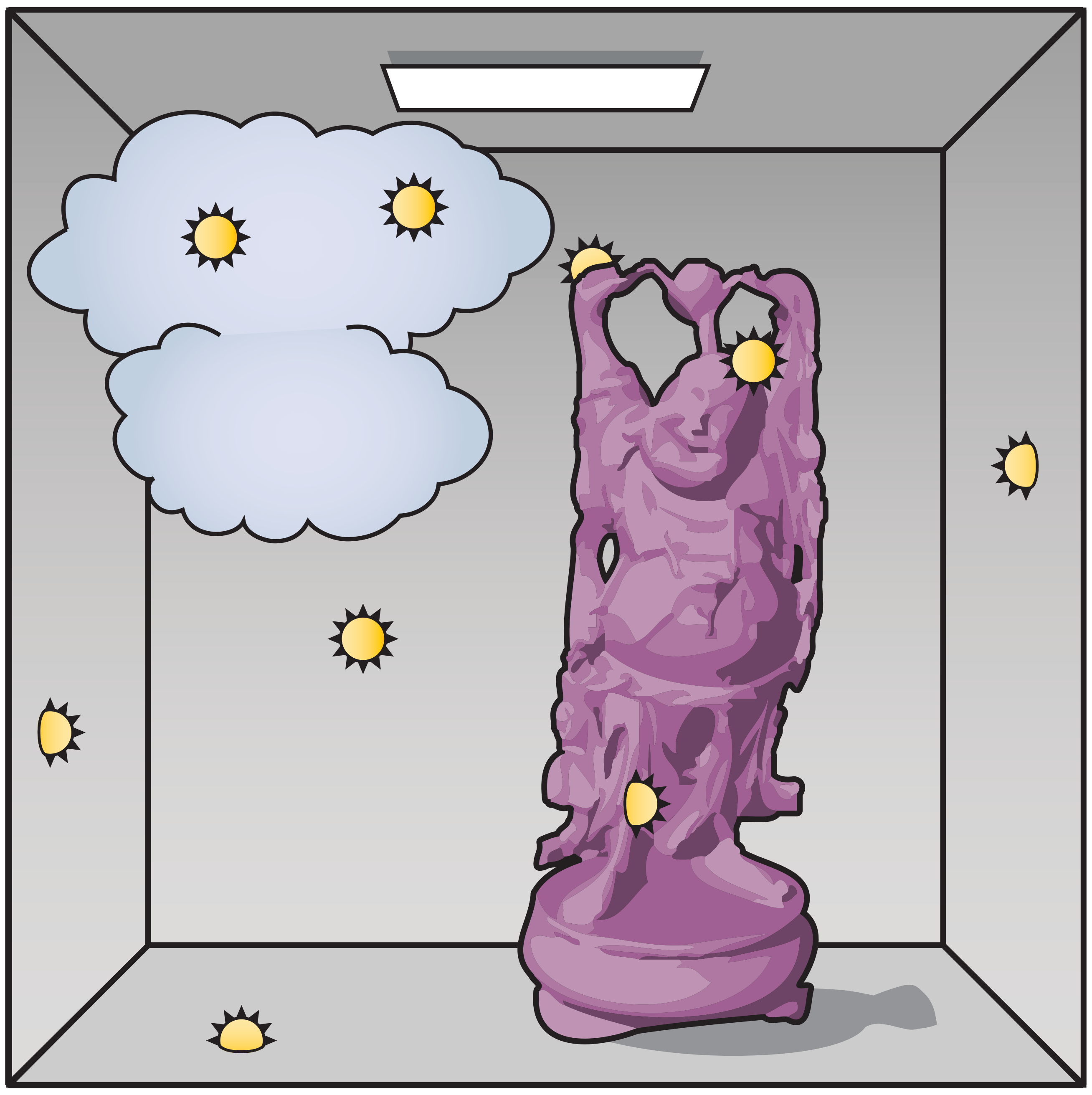


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

Pass 1: Generation of VPLs

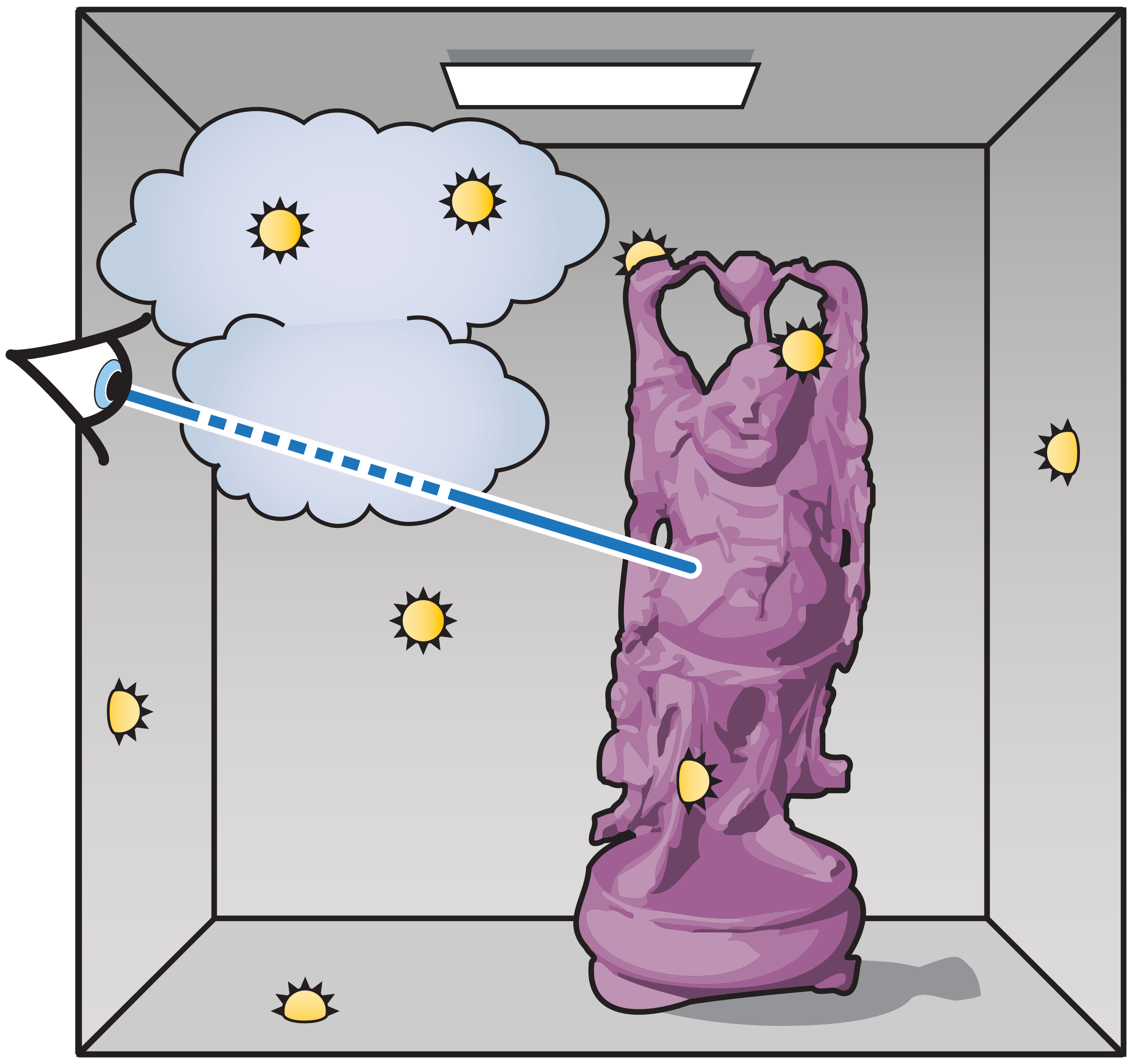


Generation of VPLs starts by sampling a position on the light source and a direction for shooting a photon.
The photon will travel along this ray until it hits the nearest surface, or as in this case scatters in the medium.

Then we sample the phase function and keep tracing the photon creating a random walk through the scene until we decide to stop and create a Virtual Point Light source. As a matter of optimization it is common to create a VPL at each bounce of the path. We often construct several such random walks to generate many Virtual Point Lights.

Many-light Methods

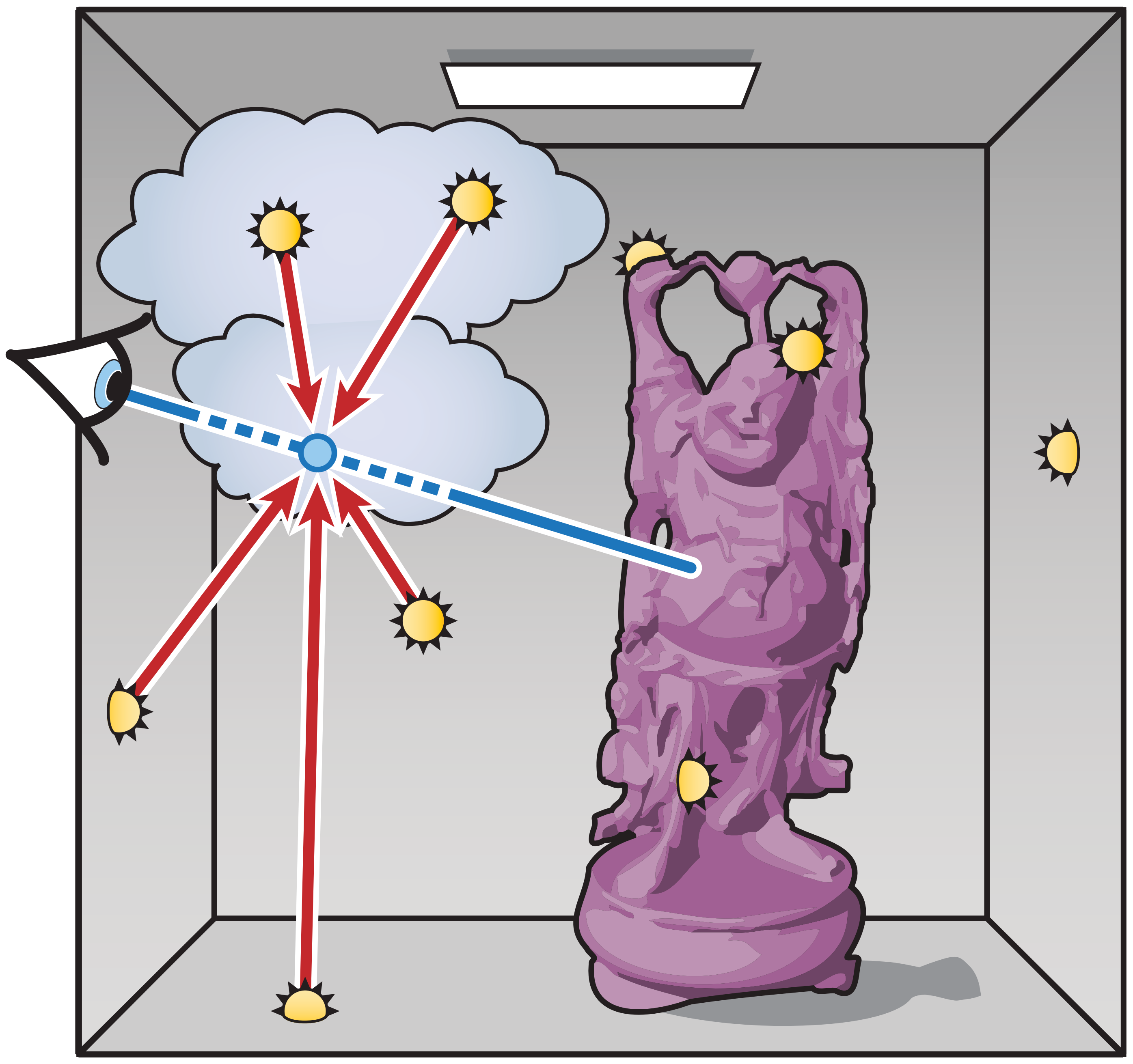
Pass 2: Lighting with VPLs



Then in the second pass, we use these VPLs to illuminate points seen by the camera... both in the media and on surfaces .

Many-light Methods

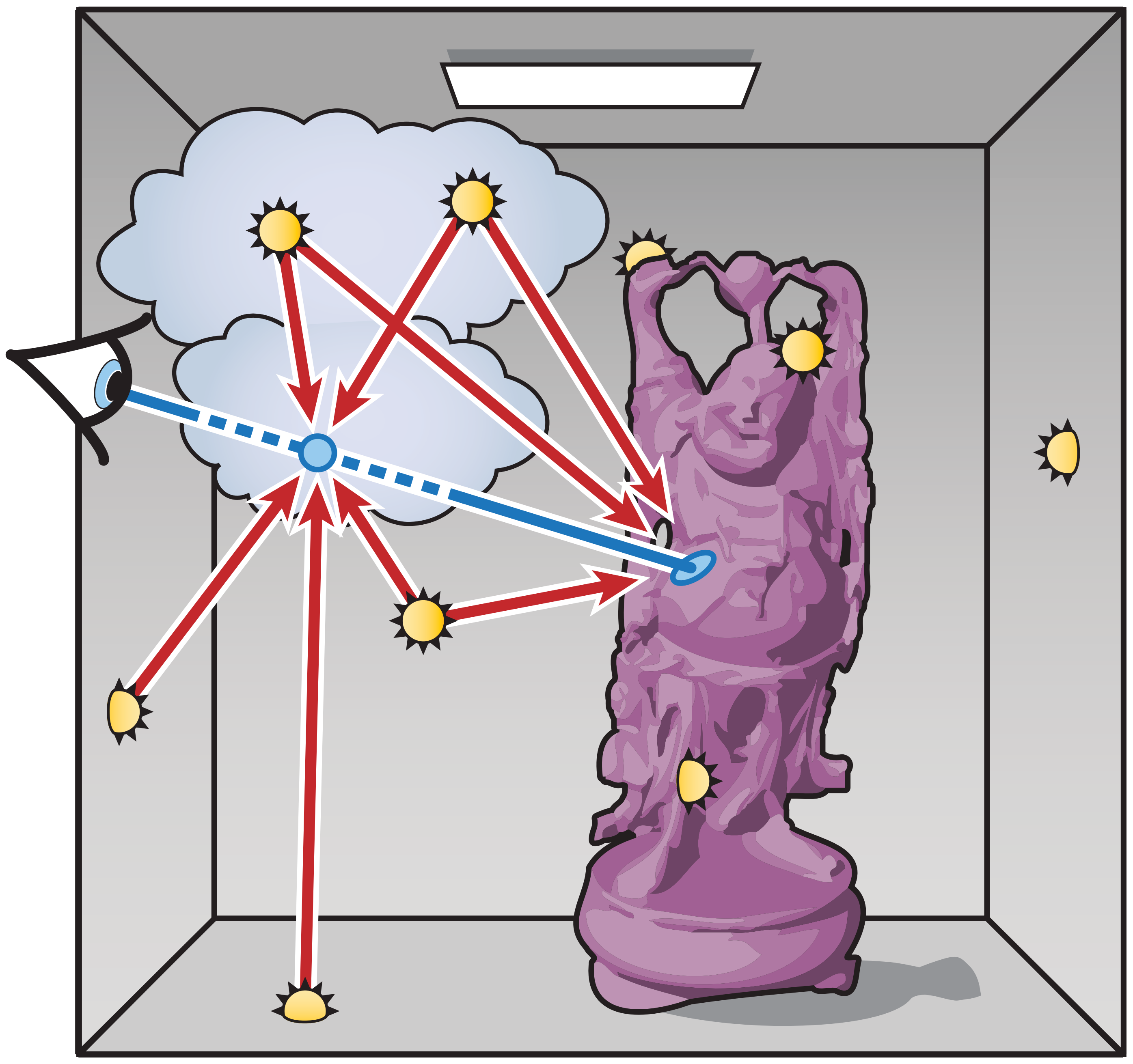
Pass 2: Lighting with VPLs



Then in the second pass, we use these VPLs to illuminate points seen by the camera... both in the media and on surfaces .

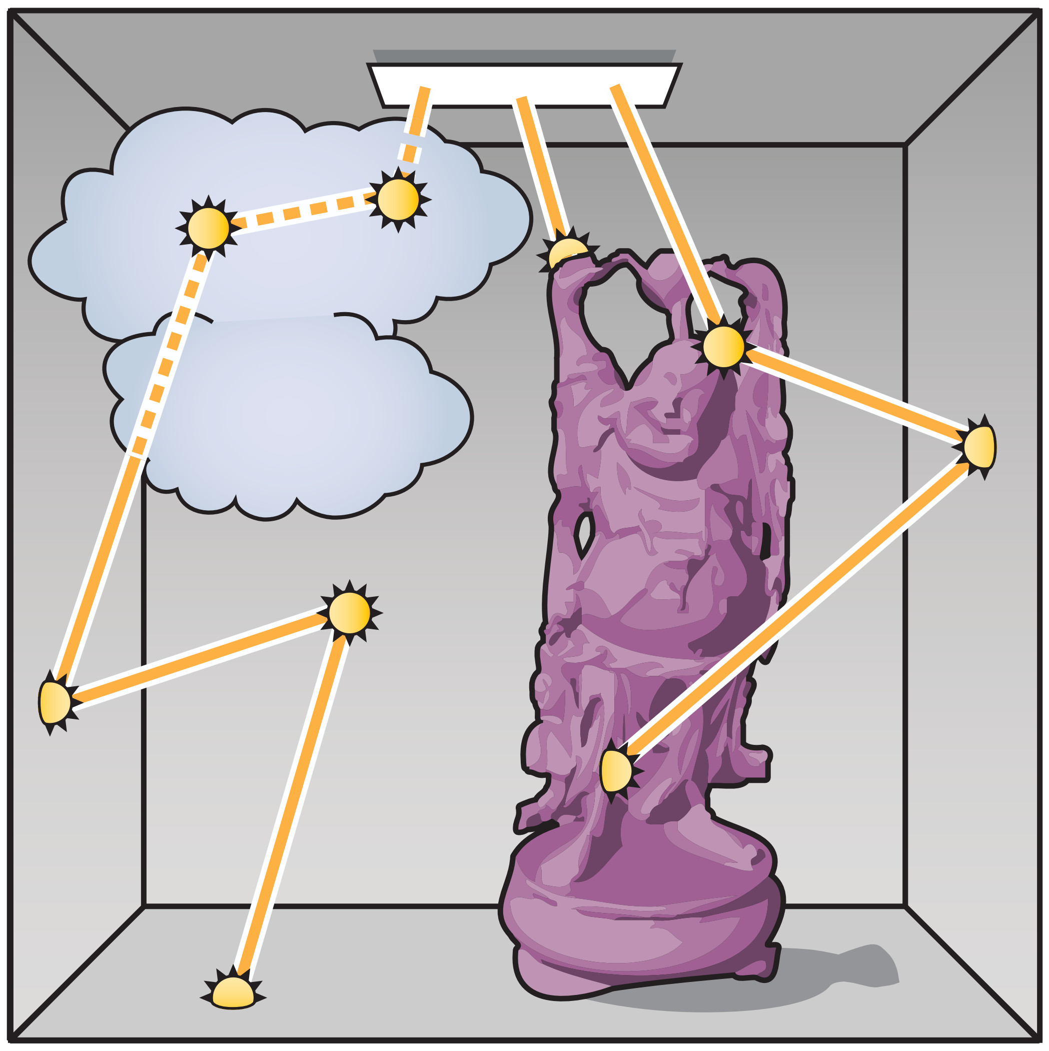
Many-light Methods

Pass 2: Lighting with VPLs

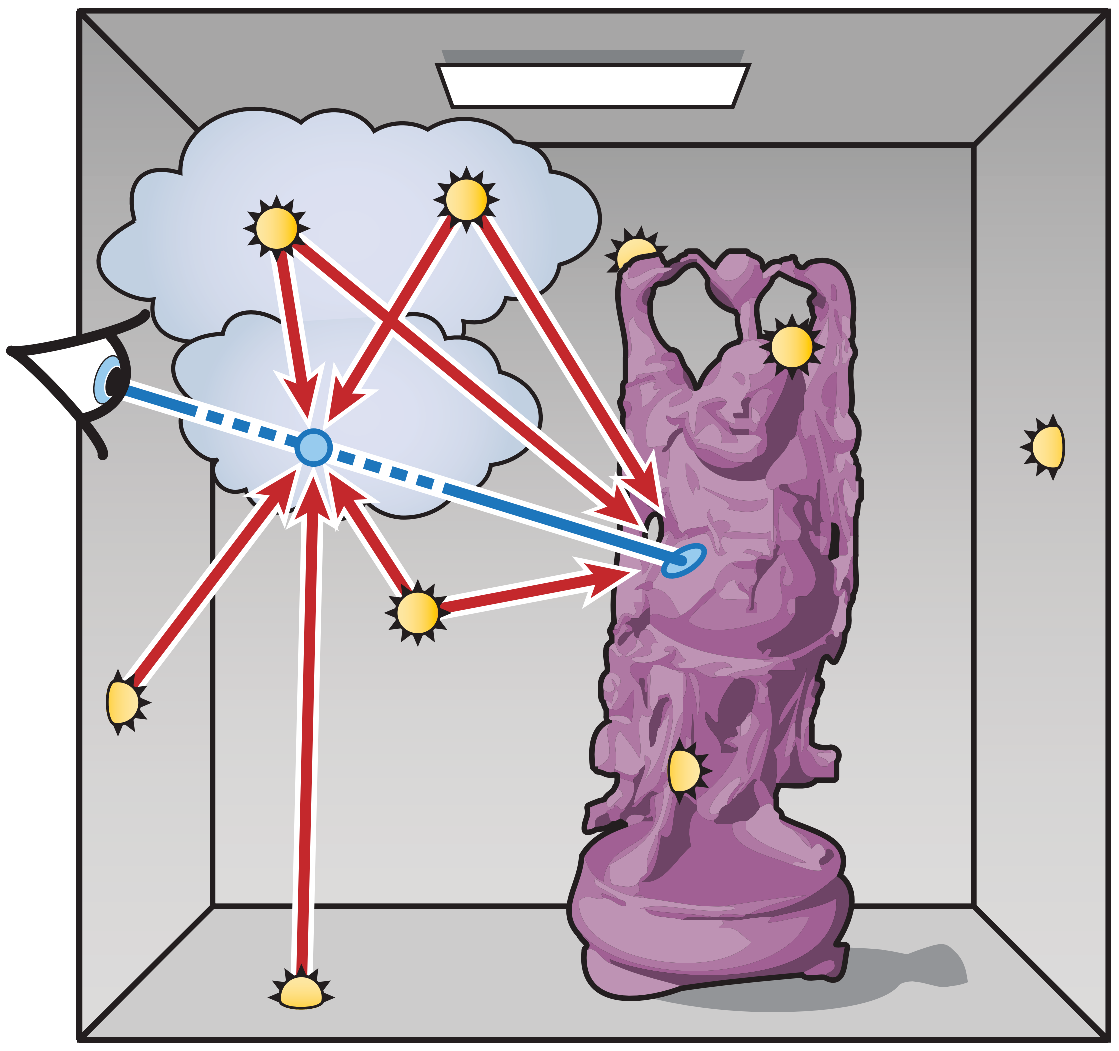


Then in the second pass, we use these VPLs to illuminate points seen by the camera... both in the media and on surfaces .

Many-light Methods



Generation of VPLs

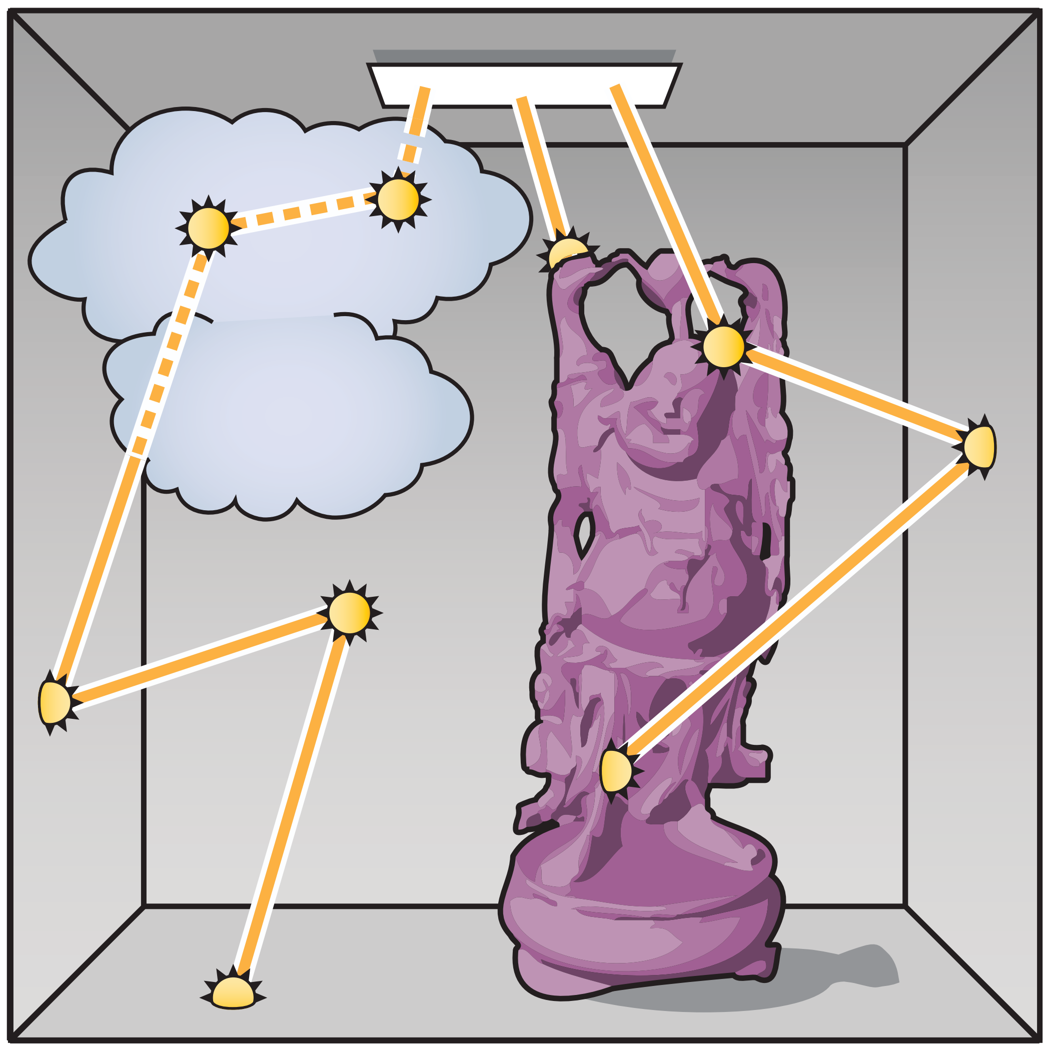


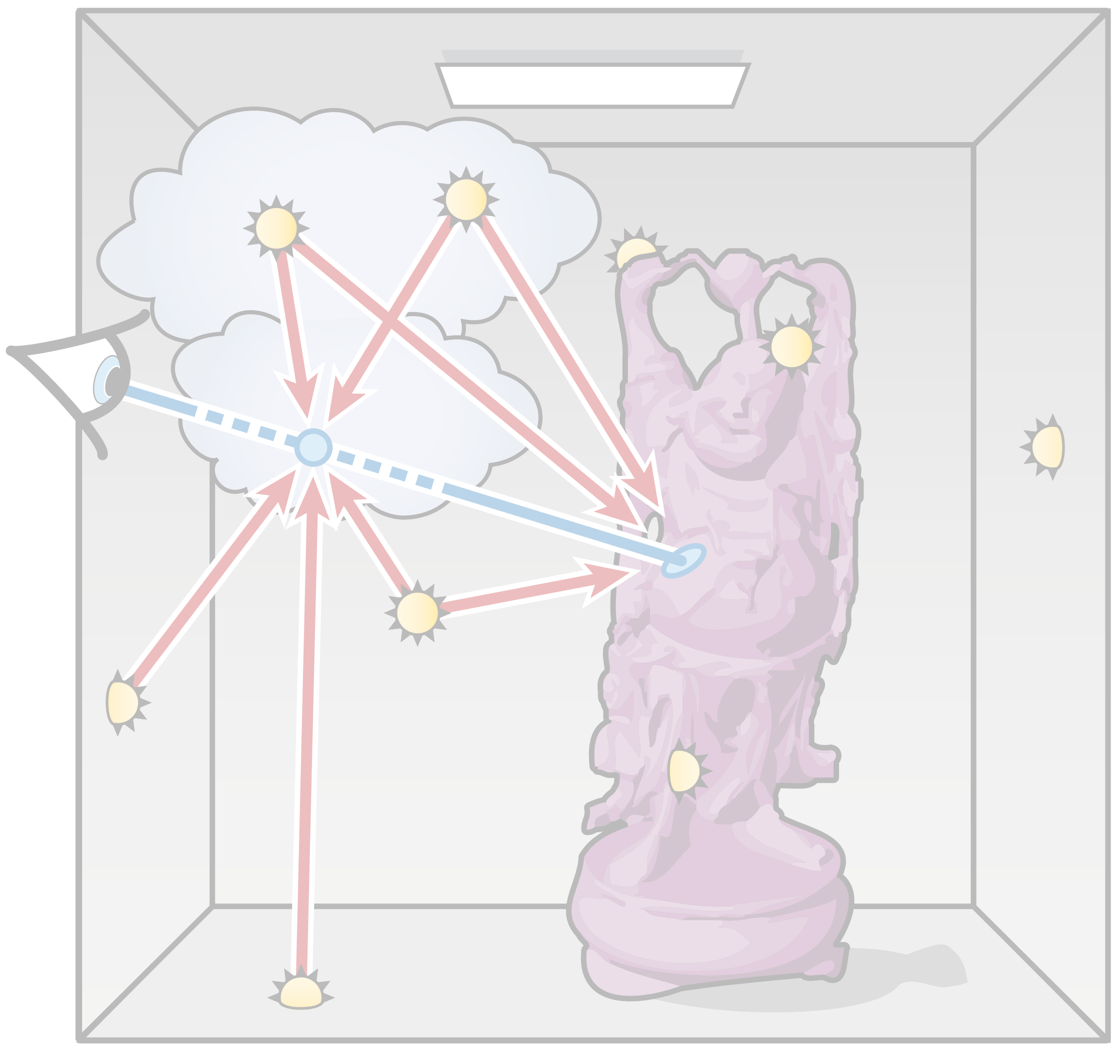
Lighting with VPLs

So, we have these two passes:

the generation of VPLs and lighting with VPLs and I will now mention the difficulties that may arise in each of them starting with the generation first.

Many-light Methods


Generation of VPLs


Lighting with VPLs

So, we have these two passes:

the generation of VPLs and lighting with VPLs and I will now mention the difficulties that may arise in each of them starting with the generation first.

Naive Generation of VPLs

Problem #1 (*in complex scenes*):

- many VPLs do not contribute

The first problem of a naive generation, such as the one that I outlined at the beginning, arises in complex scenes, where light needs to bounce several times to reach the camera.

Here we have a schematic example of such scene and you can see that most of the naively distributed VPLs will not contribute any illumination to the points seen by the camera.

Perhaps only these three.

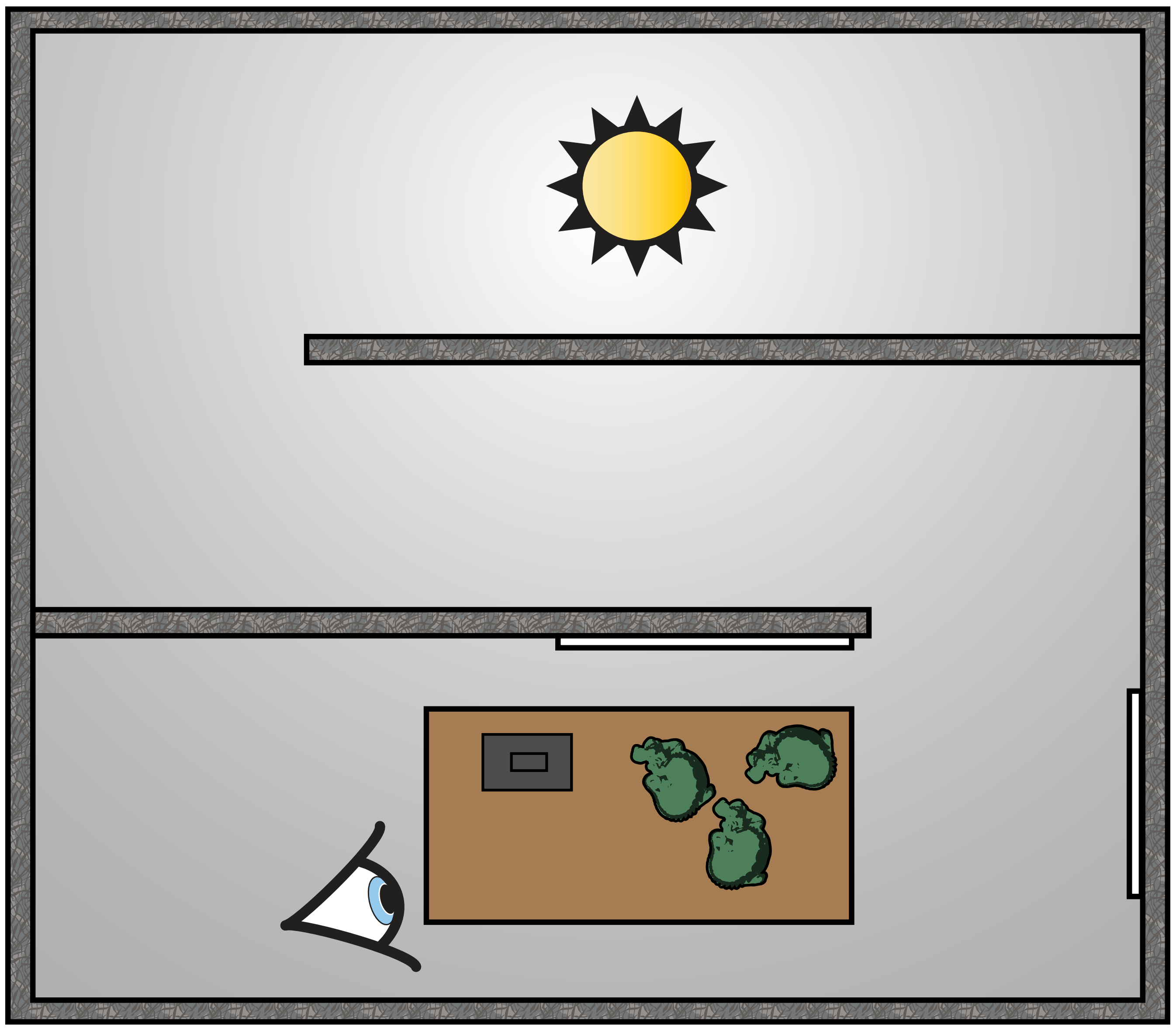
As a result we end up with a rendering that does not look plausible and may be far from the reference.

Naive Generation of VPLs

Problem #1 (*in complex scenes*):

- many VPLs do not contribute

Scene



The first problem of a naive generation, such as the one that I outlined at the beginning, arises in complex scenes, where light needs to bounce several times to reach the camera.

Here we have a schematic example of such scene and you can see that most of the naively distributed VPLs will not contribute any illumination to the points seen by the camera.

Perhaps only these three.

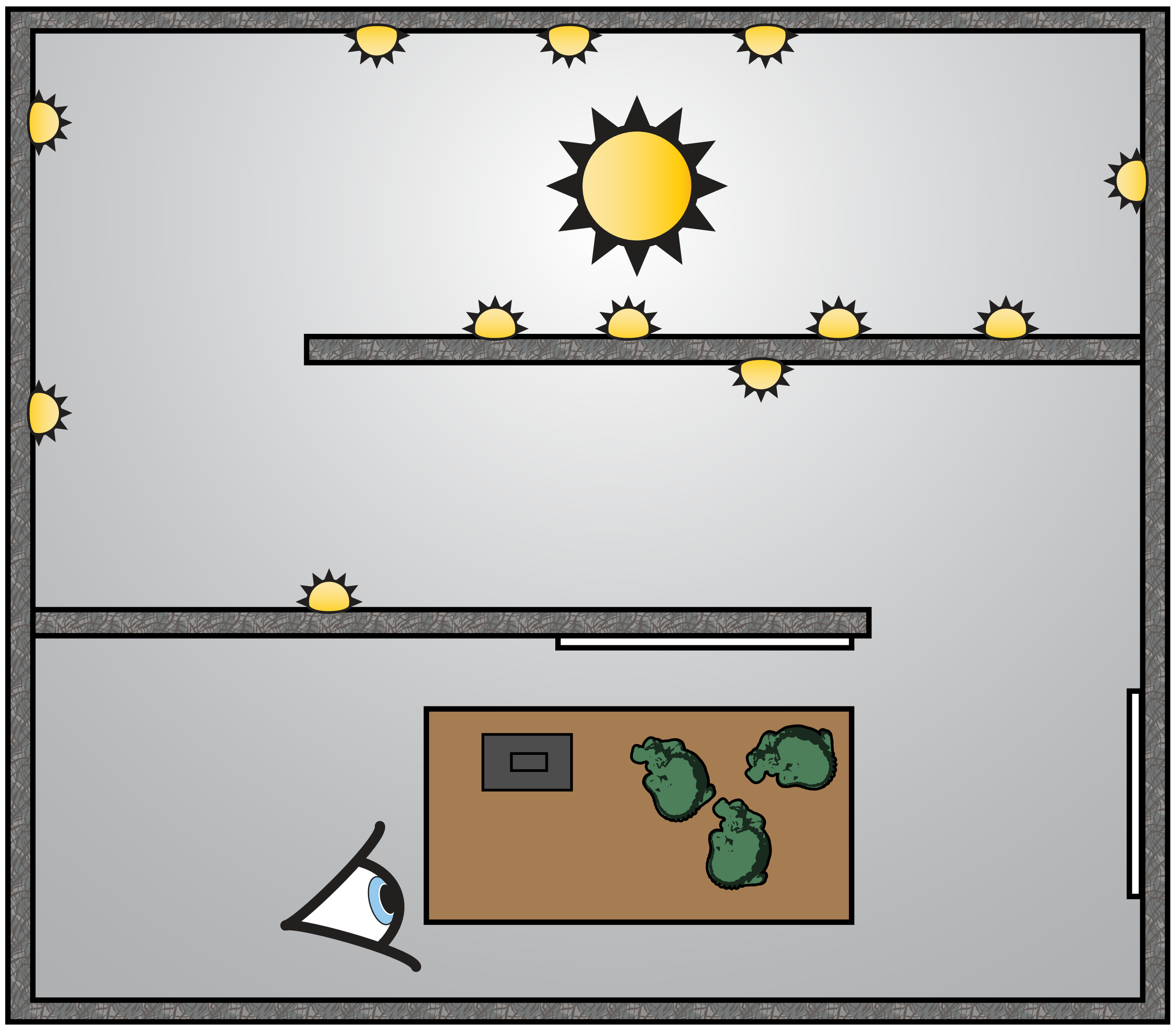
As a result we end up with a rendering that does not look plausible and may be far from the reference.

Naive Generation of VPLs

Problem #1 (*in complex scenes*):

- many VPLs do not contribute

Scene



The first problem of a naive generation, such as the one that I outlined at the beginning, arises in complex scenes, where light needs to bounce several times to reach the camera.

Here we have a schematic example of such scene and you can see that most of the naively distributed VPLs will not contribute any illumination to the points seen by the camera.

Perhaps only these three.

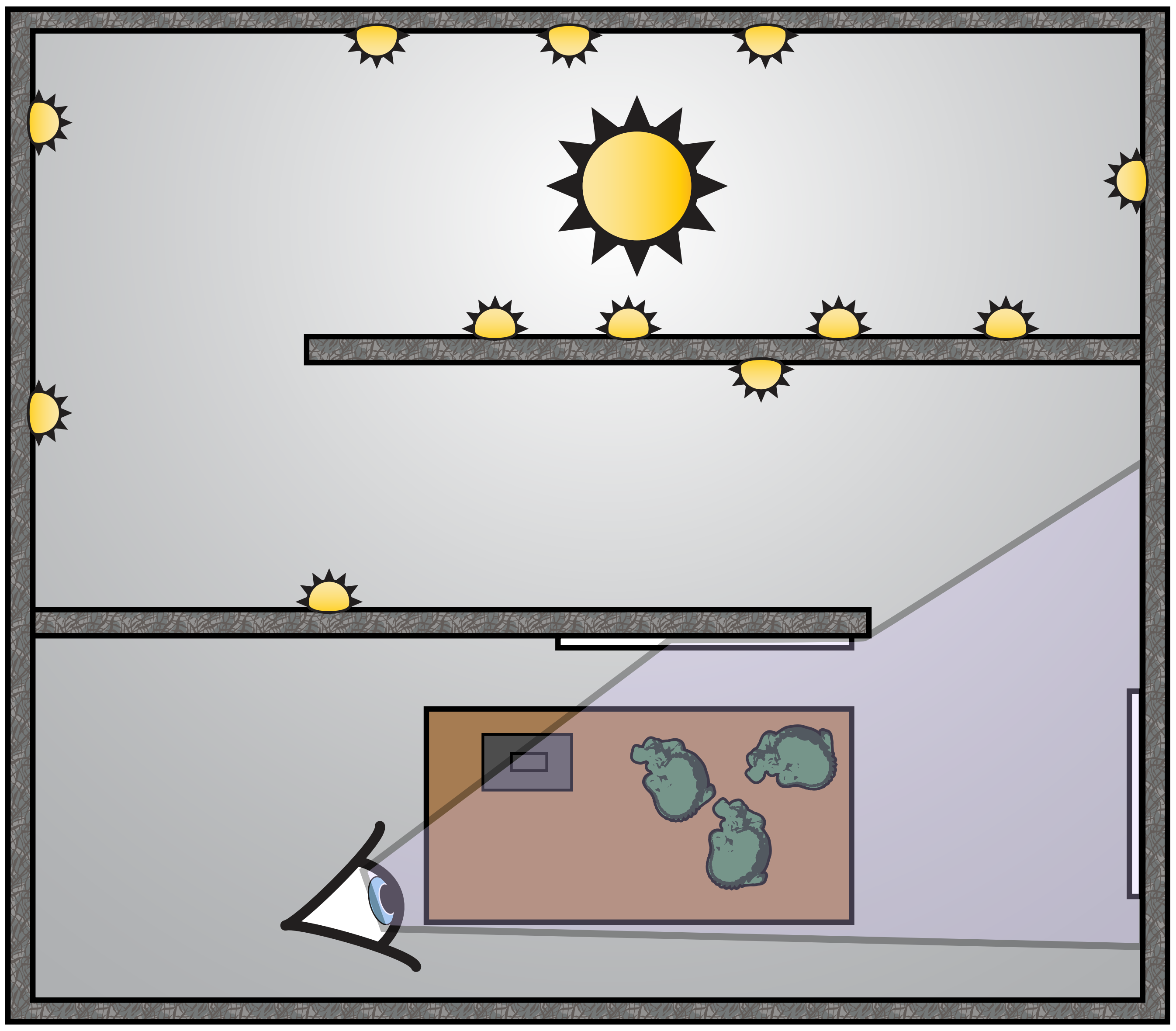
As a result we end up with a rendering that does not look plausible and may be far from the reference.

Naive Generation of VPLs

Problem #1 (*in complex scenes*):

- many VPLs do not contribute

Scene



The first problem of a naive generation, such as the one that I outlined at the beginning, arises in complex scenes, where light needs to bounce several times to reach the camera.

Here we have a schematic example of such scene and you can see that most of the naively distributed VPLs will not contribute any illumination to the points seen by the camera.

Perhaps only these three.

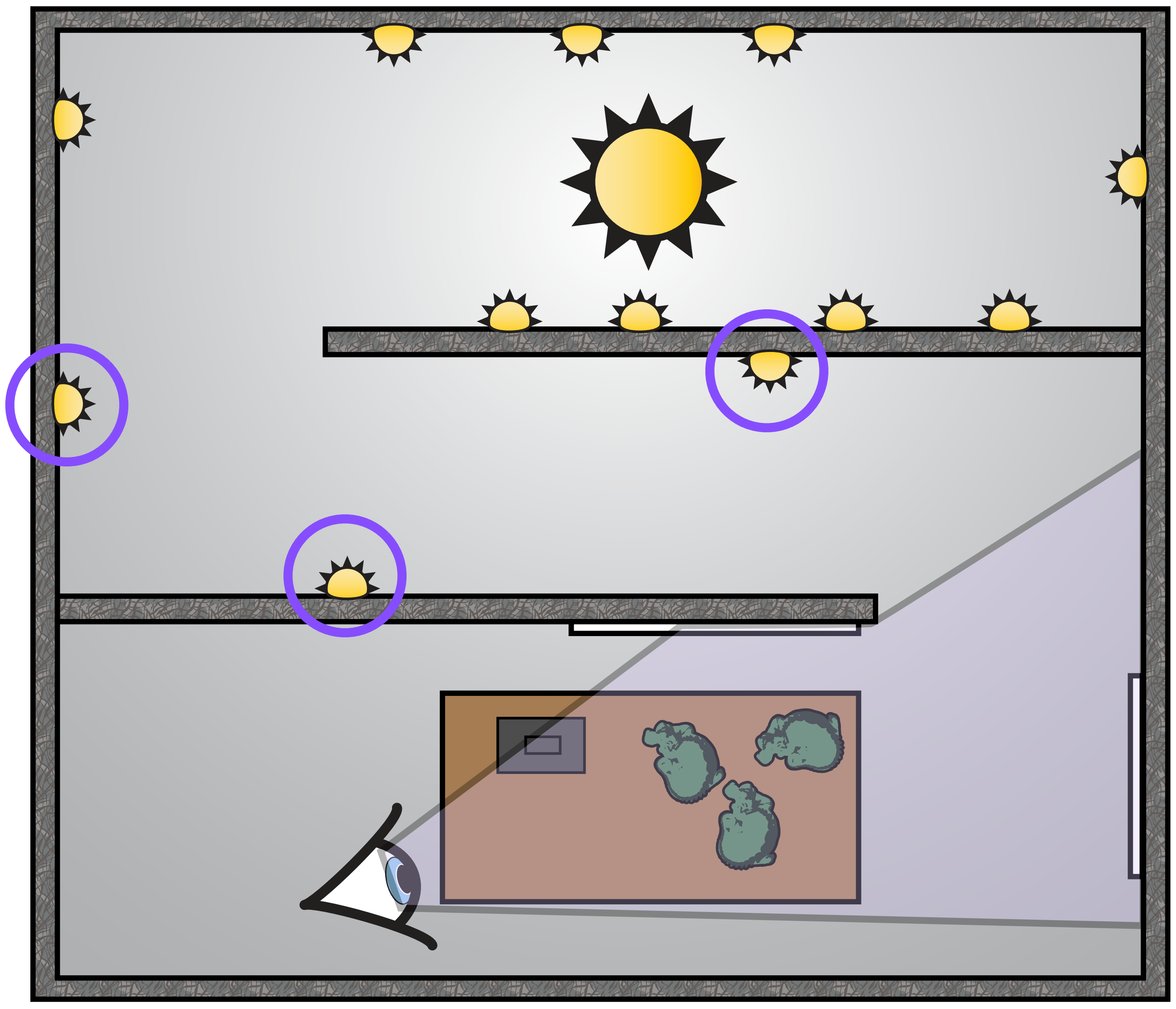
As a result we end up with a rendering that does not look plausible and may be far from the reference.

Naive Generation of VPLs

Problem #1 (*in complex scenes*):

- many VPLs do not contribute

Scene



The first problem of a naive generation, such as the one that I outlined at the beginning, arises in complex scenes, where light needs to bounce several times to reach the camera.

Here we have a schematic example of such scene and you can see that most of the naively distributed VPLs will not contribute any illumination to the points seen by the camera.

Perhaps only these three.

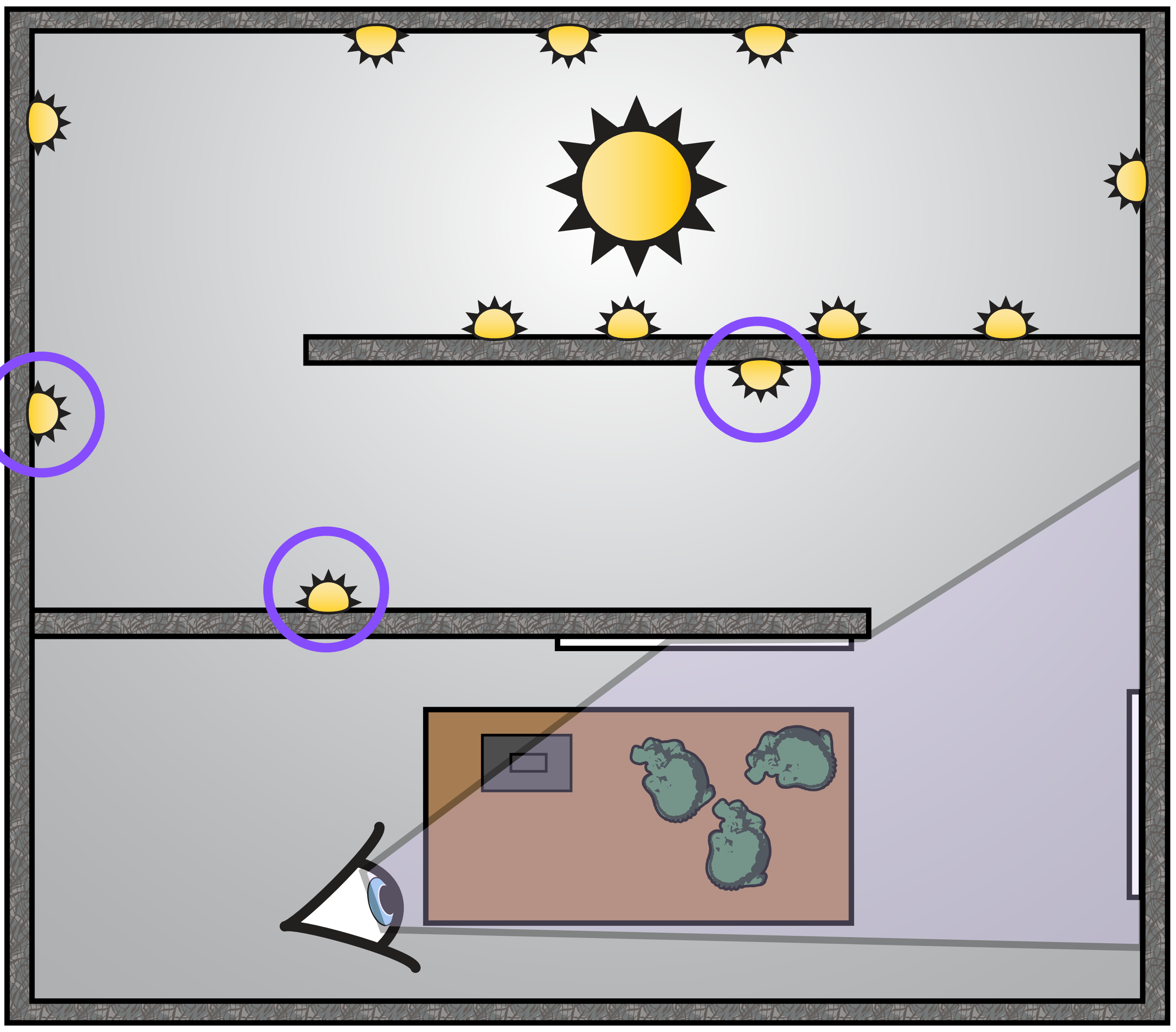
As a result we end up with a rendering that does not look plausible and may be far from the reference.

Naive Generation of VPLs

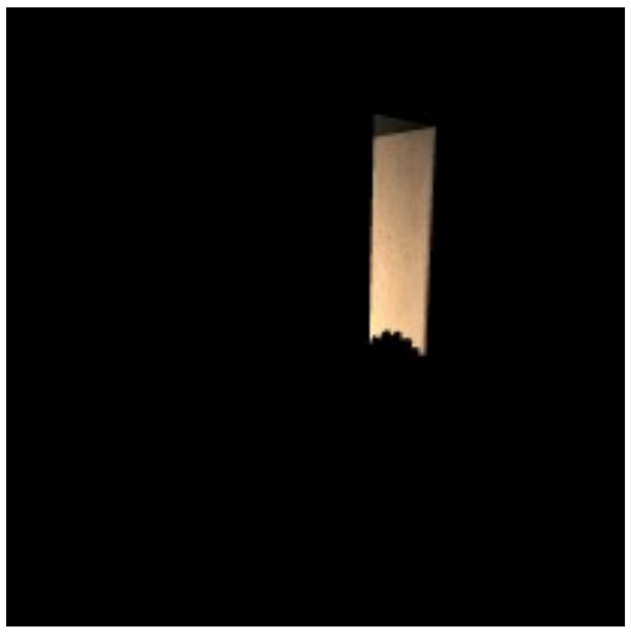
Problem #1 (*in complex scenes*):

- many VPLs do not contribute

Scene



Instant radiosity



The first problem of a naive generation, such as the one that I outlined at the beginning, arises in complex scenes, where light needs to bounce several times to reach the camera.

Here we have a schematic example of such scene and you can see that most of the naively distributed VPLs will not contribute any illumination to the points seen by the camera.

Perhaps only these three.

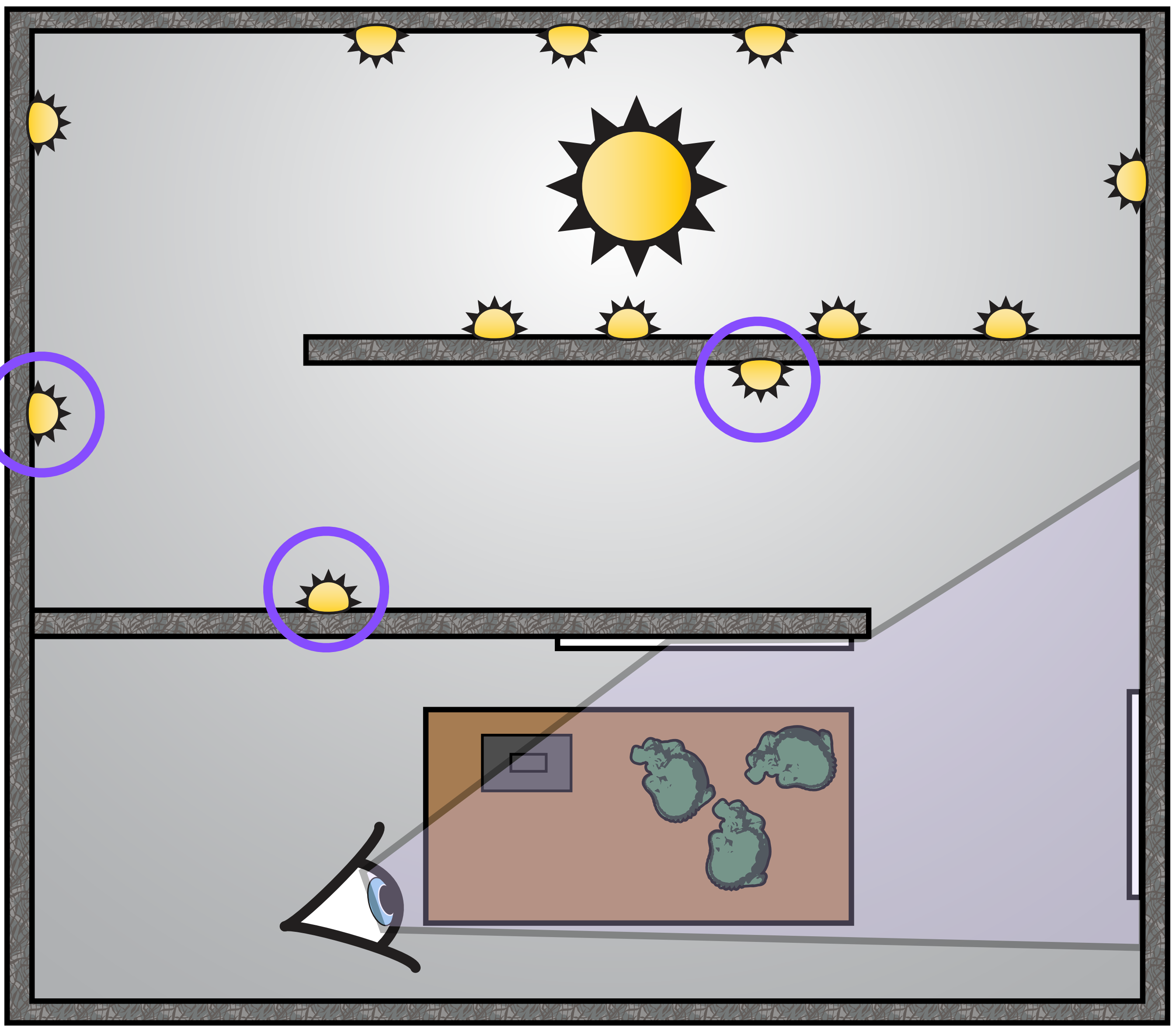
As a result we end up with a rendering that does not look plausible and may be far from the reference.

Naive Generation of VPLs

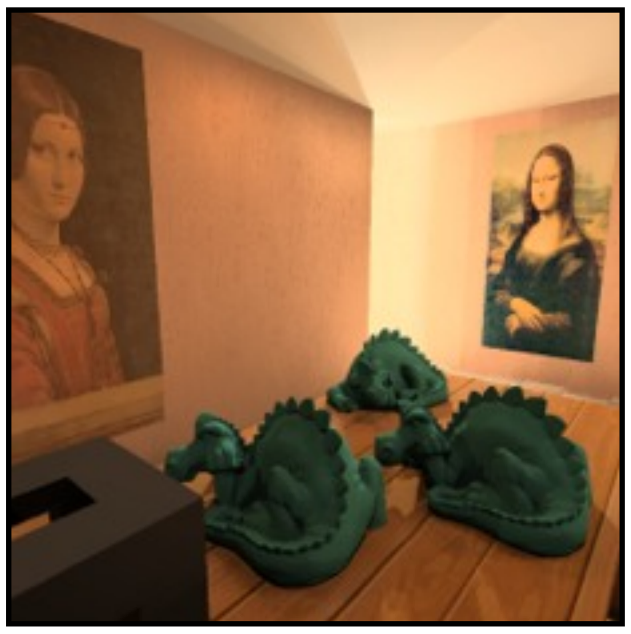
Problem #1 (*in complex scenes*):

- many VPLs do not contribute

Scene



Instant radiosity **Reference**



Images courtesy of Segovia et al.

The first problem of a naive generation, such as the one that I outlined at the beginning, arises in complex scenes, where light needs to bounce several times to reach the camera.

Here we have a schematic example of such scene and you can see that most of the naively distributed VPLs will not contribute any illumination to the points seen by the camera.

Perhaps only these three.

As a result we end up with a rendering that does not look plausible and may be far from the reference.

Naive Generation of VPLs

Another problem is that we may not have enough of VPLs in certain areas.
This happens especially in glossy scenes.

On your left you can see some artifacts, the little splotches, which emphasize the presence of the VPLs.

The common trick is to consider only the diffuse illumination and further clamp the contribution, but this significantly changes the appearance of the scene.

The reason why these artifacts are emphasized in glossy scenes is that the energy of VPLs is concentrated into narrow cones of directions, and therefore we need many VPLs to capture these reflections accurately.

Naive Generation of VPLs

Problem #2 (*in glossy scenes*):

- insufficient number of VPLs in certain areas
- glossy inter-reflections suffer from splotches

Another problem is that we may not have enough of VPLs in certain areas.
This happens especially in glossy scenes.

On your left you can see some artifacts, the little splotches, which emphasize the presence of the VPLs.

The common trick is to consider only the diffuse illumination and further clamp the contribution, but this significantly changes the appearance of the scene.

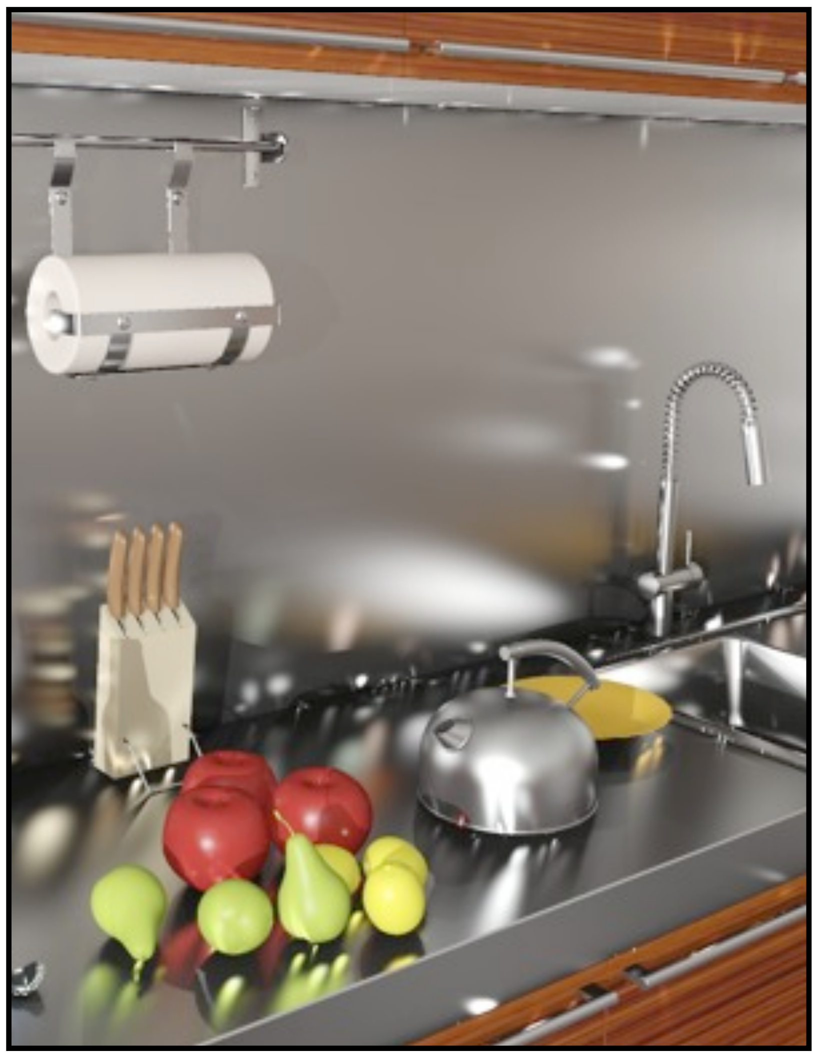
The reason why these artifacts are emphasized in glossy scenes is that the energy of VPLs is concentrated into narrow cones of directions, and therefore we need many VPLs to capture these reflections accurately.

Naive Generation of VPLs

Problem #2 (*in glossy scenes*):

- insufficient number of VPLs in certain areas
- glossy inter-reflections suffer from splotches

Instant radiosity



Another problem is that we may not have enough of VPLs in certain areas.
This happens especially in glossy scenes.

On your left you can see some artifacts, the little splotches, which emphasize the presence of the VPLs.

The common trick is to consider only the diffuse illumination and further clamp the contribution, but this significantly changes the appearance of the scene.

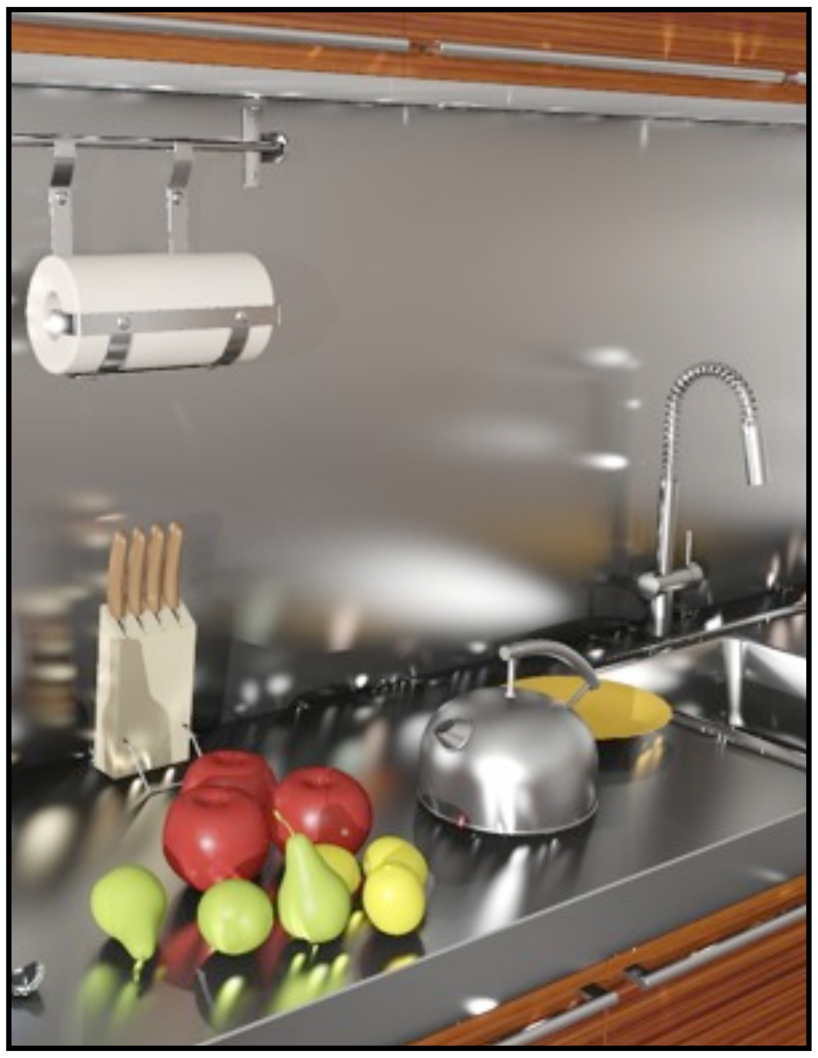
The reason why these artifacts are emphasized in glossy scenes is that the energy of VPLs is concentrated into narrow cones of directions, and therefore we need many VPLs to capture these reflections accurately.

Naive Generation of VPLs

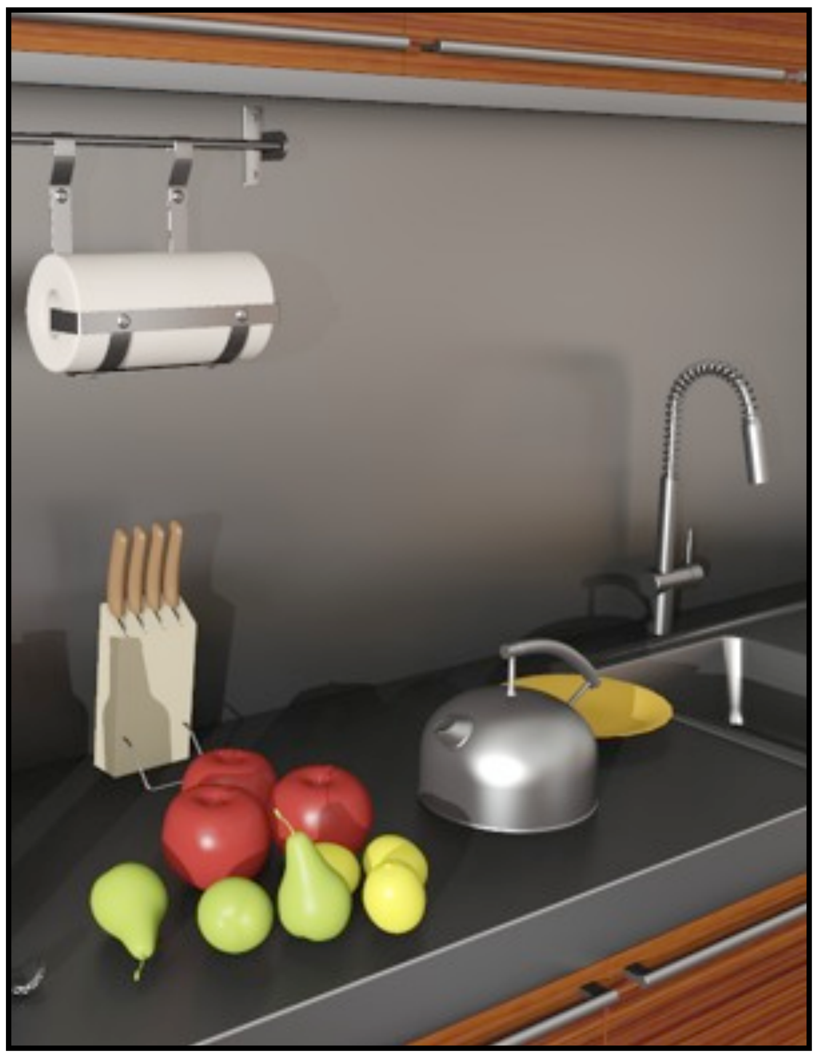
Problem #2 (*in glossy scenes*):

- insufficient number of VPLs in certain areas
- glossy inter-reflections suffer from splotches

Instant radiosity



Clamped



missing inter-reflections

Another problem is that we may not have enough of VPLs in certain areas.
This happens especially in glossy scenes.

On your left you can see some artifacts, the little splotches, which emphasize the presence of the VPLs.

The common trick is to consider only the diffuse illumination and further clamp the contribution, but this significantly changes the appearance of the scene.

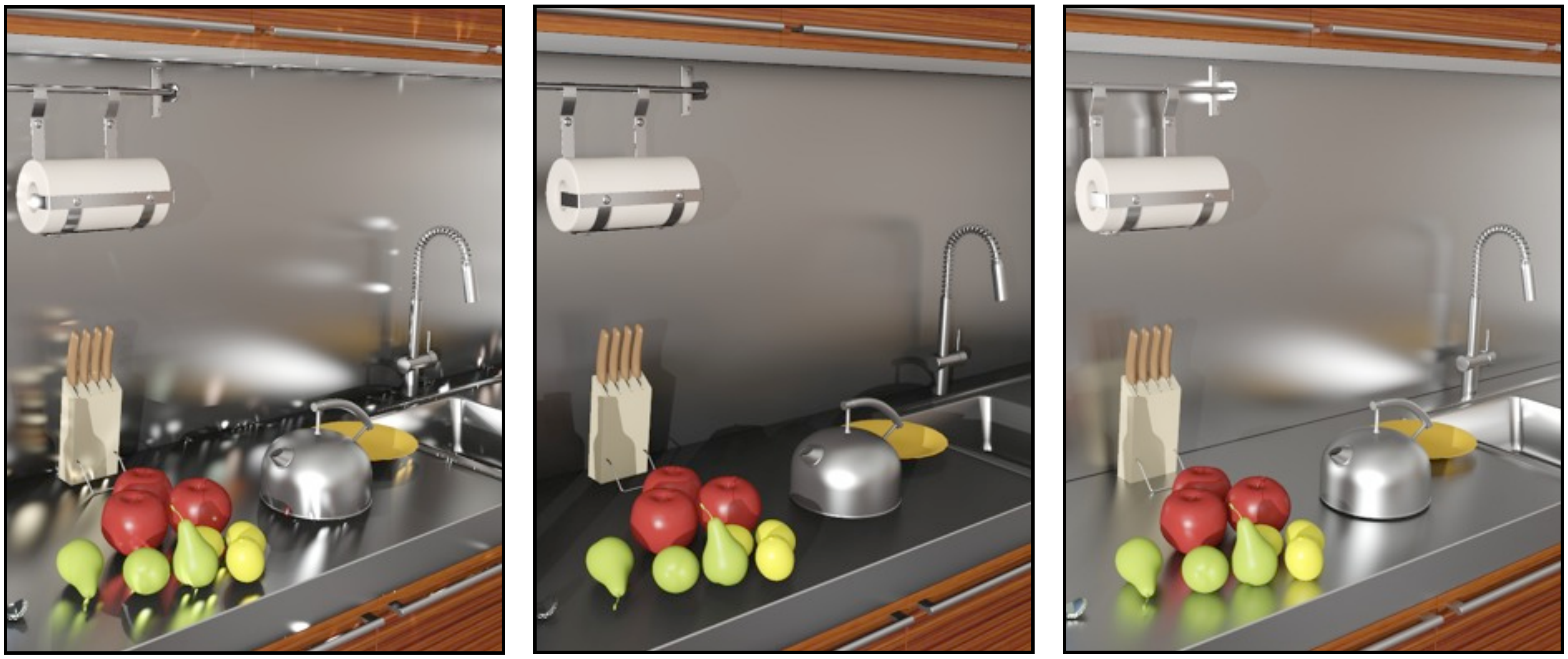
The reason why these artifacts are emphasized in glossy scenes is that the energy of VPLs is concentrated into narrow cones of directions, and therefore we need many VPLs to capture these reflections accurately.

Naive Generation of VPLs

Problem #2 (*in glossy scenes*):

- insufficient number of VPLs in certain areas
- glossy inter-reflections suffer from splotches

Instant radiosity **Clamped** **Reference**



Images courtesy of Davidovič et al.

missing inter-reflections

Another problem is that we may not have enough of VPLs in certain areas.
This happens especially in glossy scenes.

On your left you can see some artifacts, the little splotches, which emphasize the presence of the VPLs.

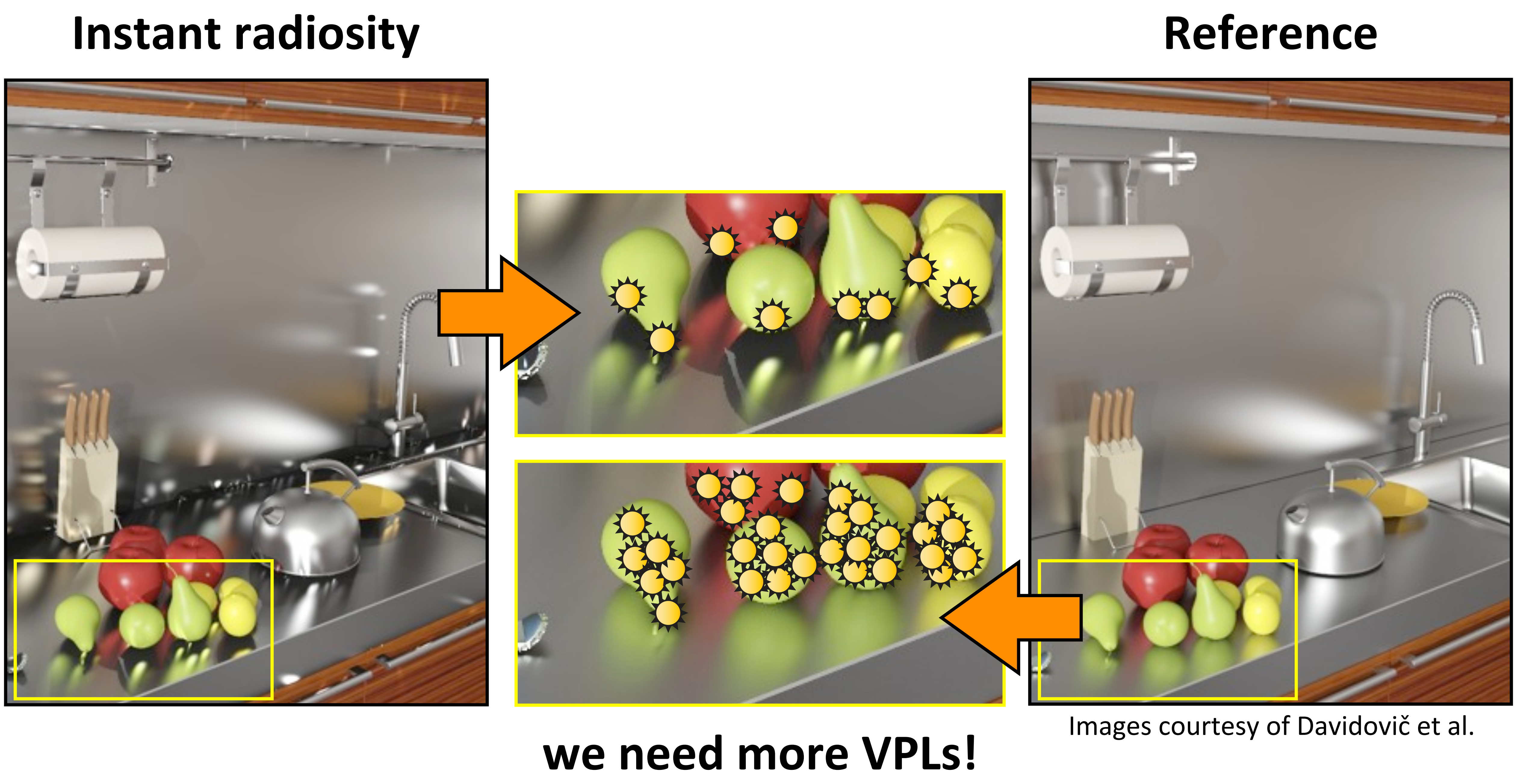
The common trick is to consider only the diffuse illumination and further clamp the contribution, but this significantly changes the appearance of the scene.

The reason why these artifacts are emphasized in glossy scenes is that the energy of VPLs is concentrated into narrow cones of directions, and therefore we need many VPLs to capture these reflections accurately.

Naive Generation of VPLs

Problem #2 (*in glossy scenes*):

- insufficient number of VPLs in certain areas
- glossy inter-reflections suffer from splotches



Images courtesy of Davidovič et al.

Another problem is that we may not have enough of VPLs in certain areas.
This happens especially in glossy scenes.

On your left you can see some artifacts, the little splotches, which emphasize the presence of the VPLs.

The common trick is to consider only the diffuse illumination and further clamp the contribution, but this significantly changes the appearance of the scene.

The reason why these artifacts are emphasized in glossy scenes is that the energy of VPLs is concentrated into narrow cones of directions, and therefore we need many VPLs to capture these reflections accurately.

Improved Generation of VPLs

So our goal is to generate VPLs only where they really need to be for the camera. There are several approaches that were proposed to achieve this.

The most straightforward one is to simply reject VPLs that are not likely to contribute, other techniques use bidirectional and metropolis sampling to create VPLs. The last one focuses on highly glossy scenes creating local virtual lights to compensate for the energy loss due to clamping.

Improved Generation of VPLs

Goal:

- place VPLs only where needed

So our goal is to generate VPLs only where they really need to be for the camera. There are several approaches that were proposed to achieve this.

The most straightforward one is to simply reject VPLs that are not likely to contribute, other techniques use bidirectional and metropolis sampling to create VPLs. The last one focuses on highly glossy scenes creating local virtual lights to compensate for the energy loss due to clamping.

Improved Generation of VPLs

Goal:

- place VPLs only where needed

Approaches:

- Rejection of unimportant VPLs [Georgiev and Slusallek 2010]
- Bidirectional Instant Radiosity [Segovia et al. 2006]
- Metropolis sampling for VPL distributions [Segovia et al. 2007]
- Local Virtual Lights [Davidovič et al. 2010]

So our goal is to generate VPLs only where they really need to be for the camera. There are several approaches that were proposed to achieve this.

The most straightforward one is to simply reject VPLs that are not likely to contribute, other techniques use bidirectional and metropolis sampling to create VPLs. The last one focuses on highly glossy scenes creating local virtual lights to compensate for the energy loss due to clamping.

Improved Generation of VPLs

The first technique is a form of rejection sampling and was presented in a short paper by Georgiev and Slusallek at Eurographics 2010.

The idea is very simple, they use the naive generation algorithm but probabilistically reject VPLs whose contribution is expected to be less than on average.

This way they end up keeping only few VPLs that will contribute almost equally.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

The first technique is a form of rejection sampling and was presented in a short paper by Georgiev and Slusallek at Eurographics 2010.

The idea is very simple, they use the naive generation algorithm but probabilistically reject VPLs whose contribution is expected to be less than on average.

This way they end up keeping only few VPLs that will contribute almost equally.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Idea:

- probabilistically reject VPLs with low expected contribution

The first technique is a form of rejection sampling and was presented in a short paper by Georgiev and Slusallek at Eurographics 2010.

The idea is very simple, they use the naive generation algorithm but probabilistically reject VPLs whose contribution is expected to be less than on average.

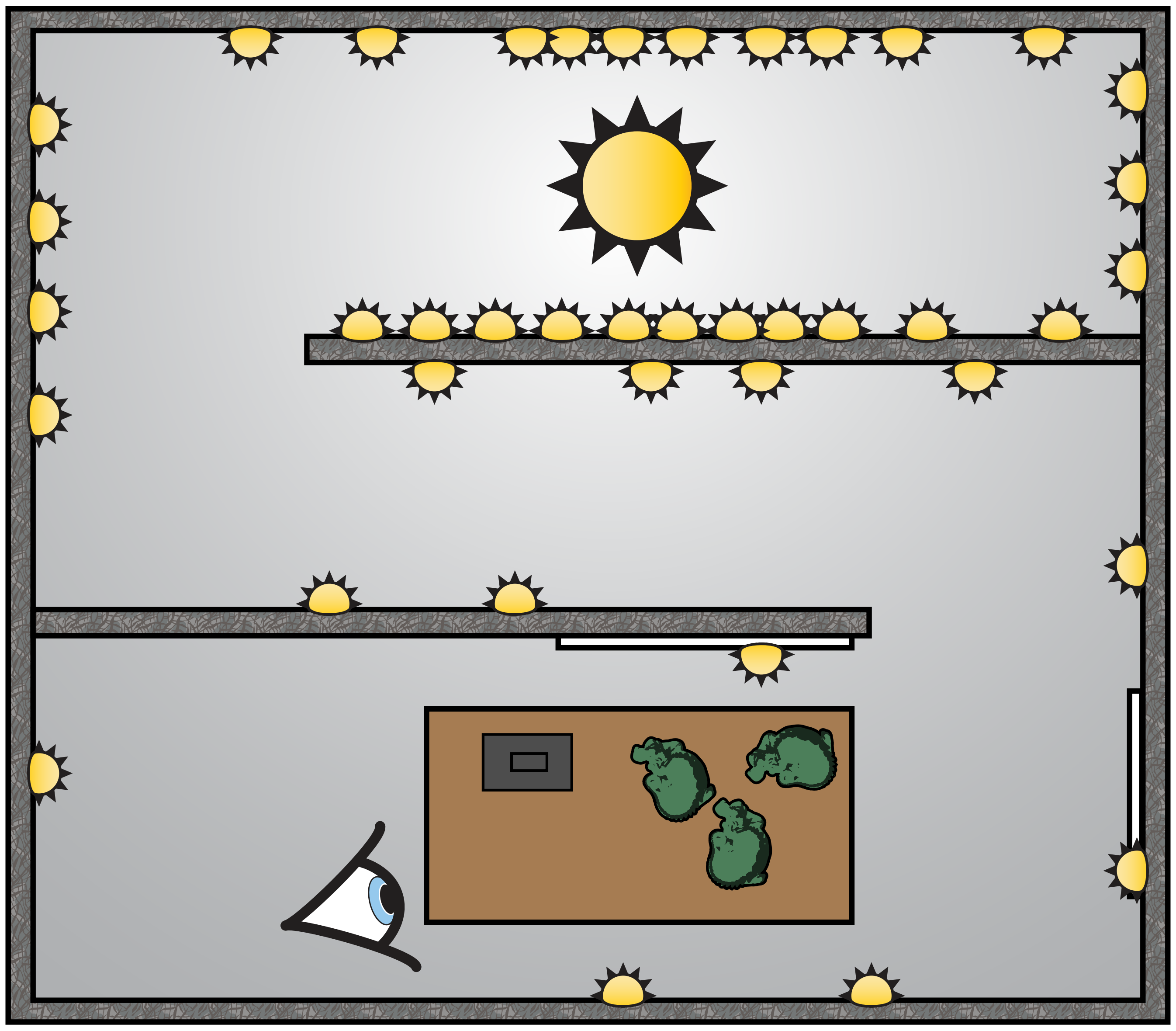
This way they end up keeping only few VPLs that will contribute almost equally.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Idea:

- probabilistically reject VPLs with low expected contribution



The first technique is a form of rejection sampling and was presented in a short paper by Georgiev and Slusallek at Eurographics 2010.

The idea is very simple, they use the naive generation algorithm but probabilistically reject VPLs whose contribution is expected to be less than on average.

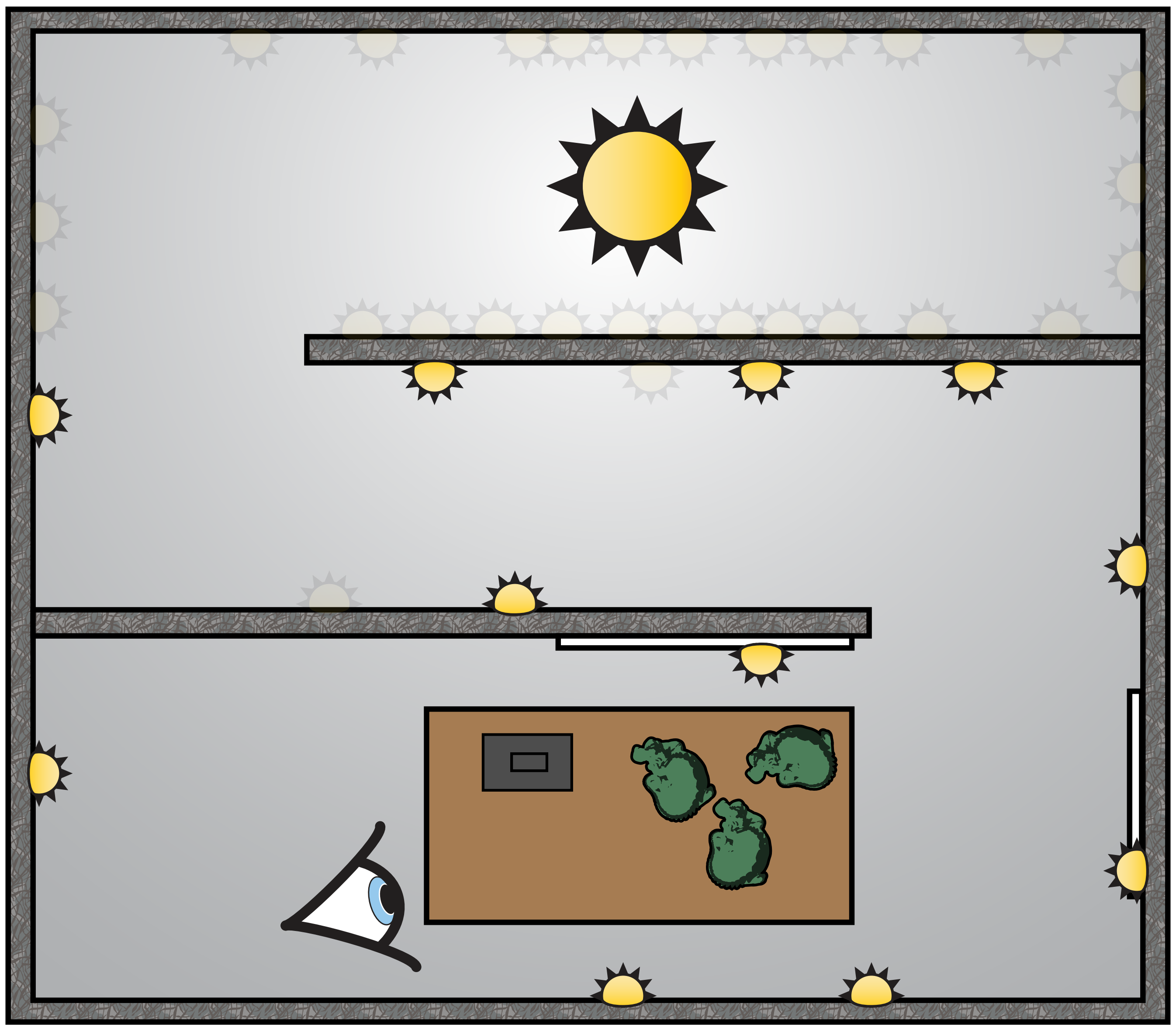
This way they end up keeping only few VPLs that will contribute almost equally.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Idea:

- probabilistically reject VPLs with low expected contribution



The first technique is a form of rejection sampling and was presented in a short paper by Georgiev and Slusallek at Eurographics 2010.

The idea is very simple, they use the naive generation algorithm but probabilistically reject VPLs whose contribution is expected to be less than on average.

This way they end up keeping only few VPLs that will contribute almost equally.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Approach:

The algorithm works as follows:

First they estimate the average contribution of a VPL Phi_v. For this they shoot a few pilot VPLs and compute the average contribution to few surface points seen by the camera.

Then when generating the VPLs, they take each created one, estimate its contribution, again by computing the illumination of few surface points seen by the camera.

The VPL is kept with probability that is proportional to its relative contribution with respect to the average.
To account for the rejected VPLs, they simply divide by the acceptance probability

The last two steps are basically Russian roulette driven by the expected contribution to the image.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Approach:

1. estimate average contribution of a VPL Φ_v

The algorithm works as follows:

First they estimate the average contribution of a VPL Phi_v. For this they shoot a few pilot VPLs and compute the average contribution to few surface points seen by the camera.

Then when generating the VPLs, they take each created one, estimate its contribution, again by computing the illumination of few surface points seen by the camera.

The VPL is kept with probability that is proportional to its relative contribution with respect to the average.
To account for the rejected VPLs, they simply divide by the acceptance probability

The last two steps are basically Russian roulette driven by the expected contribution to the image.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Approach:

1. estimate average contribution of a VPL Φ_v
   - few pilot VPLs illuminate few surface points seen by the camera

The algorithm works as follows:

First they estimate the average contribution of a VPL Phi_v. For this they shoot a few pilot VPLs and compute the average contribution to few surface points seen by the camera.

Then when generating the VPLs, they take each created one, estimate its contribution, again by computing the illumination of few surface points seen by the camera.

The VPL is kept with probability that is proportional to its relative contribution with respect to the average.
To account for the rejected VPLs, they simply divide by the acceptance probability

The last two steps are basically Russian roulette driven by the expected contribution to the image.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Approach:

1. estimate average contribution of a VPL Φ_v
   - few pilot VPLs illuminate few surface points seen by the camera
2. generate VPLs
   - for each VPL i
     - estimate its contribution Φ_i to points seen by the camera

The algorithm works as follows:

First they estimate the average contribution of a VPL Phi_v. For this they shoot a few pilot VPLs and compute the average contribution to few surface points seen by the camera.

Then when generating the VPLs, they take each created one, estimate its contribution, again by computing the illumination of few surface points seen by the camera.

The VPL is kept with probability that is proportional to its relative contribution with respect to the average.
To account for the rejected VPLs, they simply divide by the acceptance probability

The last two steps are basically Russian roulette driven by the expected contribution to the image.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Approach:

1. estimate average contribution of a VPL Φ_v
   - few pilot VPLs illuminate few surface points seen by the camera
2. generate VPLs
   - for each VPL i
     - estimate its contribution Φ_i to points seen by the camera
     - accept with probability $p_i = \min\left(\frac{\Phi_i}{\Phi_v} + \epsilon, 1\right)$

The algorithm works as follows:

First they estimate the average contribution of a VPL Phi_v. For this they shoot a few pilot VPLs and compute the average contribution to few surface points seen by the camera.

Then when generating the VPLs, they take each created one, estimate its contribution, again by computing the illumination of few surface points seen by the camera.

The VPL is kept with probability that is proportional to its relative contribution with respect to the average.
To account for the rejected VPLs, they simply divide by the acceptance probability

The last two steps are basically Russian roulette driven by the expected contribution to the image.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Approach:

1. estimate average contribution of a VPL Φ_v
   - few pilot VPLs illuminate few surface points seen by the camera

2. generate VPLs
   - for each VPL i
     - estimate its contribution Φ_i to points seen by the camera
     - accept with probability $p_i = \min\left(\frac{\Phi_i}{\Phi_v} + \epsilon, 1\right)$
     - if accepted, divide its energy by p_i

The algorithm works as follows:

First they estimate the average contribution of a VPL Phi_v. For this they shoot a few pilot VPLs and compute the average contribution to few surface points seen by the camera.

Then when generating the VPLs, they take each created one, estimate its contribution, again by computing the illumination of few surface points seen by the camera.

The VPL is kept with probability that is proportional to its relative contribution with respect to the average.
To account for the rejected VPLs, they simply divide by the acceptance probability

The last two steps are basically Russian roulette driven by the expected contribution to the image.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Approach:

1. estimate average contribution of a VPL Φ_v
   - few pilot VPLs illuminate few surface points seen by the camera

2. generate VPLs
   - for each VPL i
     - estimate its contribution Φ_i to points seen by the camera
     - accept with probability $p_i = \min\left(\frac{\Phi_i}{\Phi_v} + \epsilon, 1\right)$
     - if accepted, divide its energy by p_i

Russian roulette

The algorithm works as follows:

First they estimate the average contribution of a VPL Phi_v. For this they shoot a few pilot VPLs and compute the average contribution to few surface points seen by the camera.

Then when generating the VPLs, they take each created one, estimate its contribution, again by computing the illumination of few surface points seen by the camera.

The VPL is kept with probability that is proportional to its relative contribution with respect to the average.
To account for the rejected VPLs, they simply divide by the acceptance probability

The last two steps are basically Russian roulette driven by the expected contribution to the image.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Advantages:

- cheap and simple to implement!
- VPLs have roughly equal contribution

The advantages of rejection sampling are the simplicity and that the VPLs have roughly equal contribution.

The drawbacks are the increased cost of the VPL distribution pass.

and also the one-pixel image assumption, which means that we do not create different VPLs for different pixels
and thus this technique does not help much with the problem of glossy inter-reflections.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Advantages:

- cheap and simple to implement!
- VPLs have roughly equal contribution
- works well most of the time

The advantages of rejection sampling are the simplicity and that the VPLs have roughly equal contribution.

The drawbacks are the increased cost of the VPL distribution pass.

and also the one-pixel image assumption, which means that we do not create different VPLs for different pixels
and thus this technique does not help much with the problem of glossy inter-reflections.

Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Advantages:

- cheap and simple to implement!
- VPLs have roughly equal contribution
- works well most of the time

Disadvantages:

- increases the cost of VPL distribution
- "one-pixel image" assumption
- does not help with local inter-reflections

The advantages of rejection sampling are the simplicity and that the VPLs have roughly equal contribution.

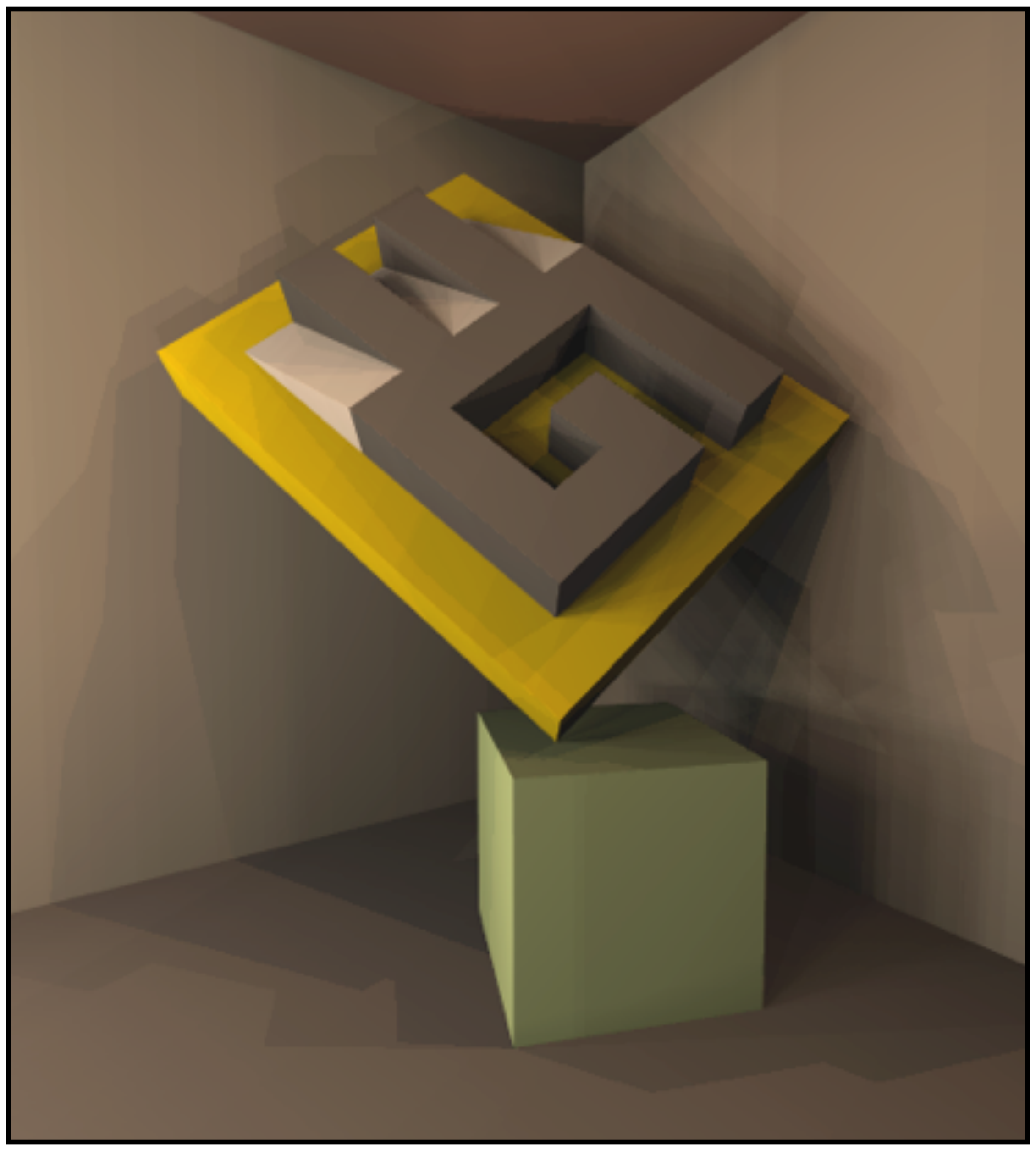
The drawbacks are the increased cost of the VPL distribution pass.

and also the one-pixel image assumption, which means that we do not create different VPLs for different pixels
and thus this technique does not help much with the problem of glossy inter-reflections.

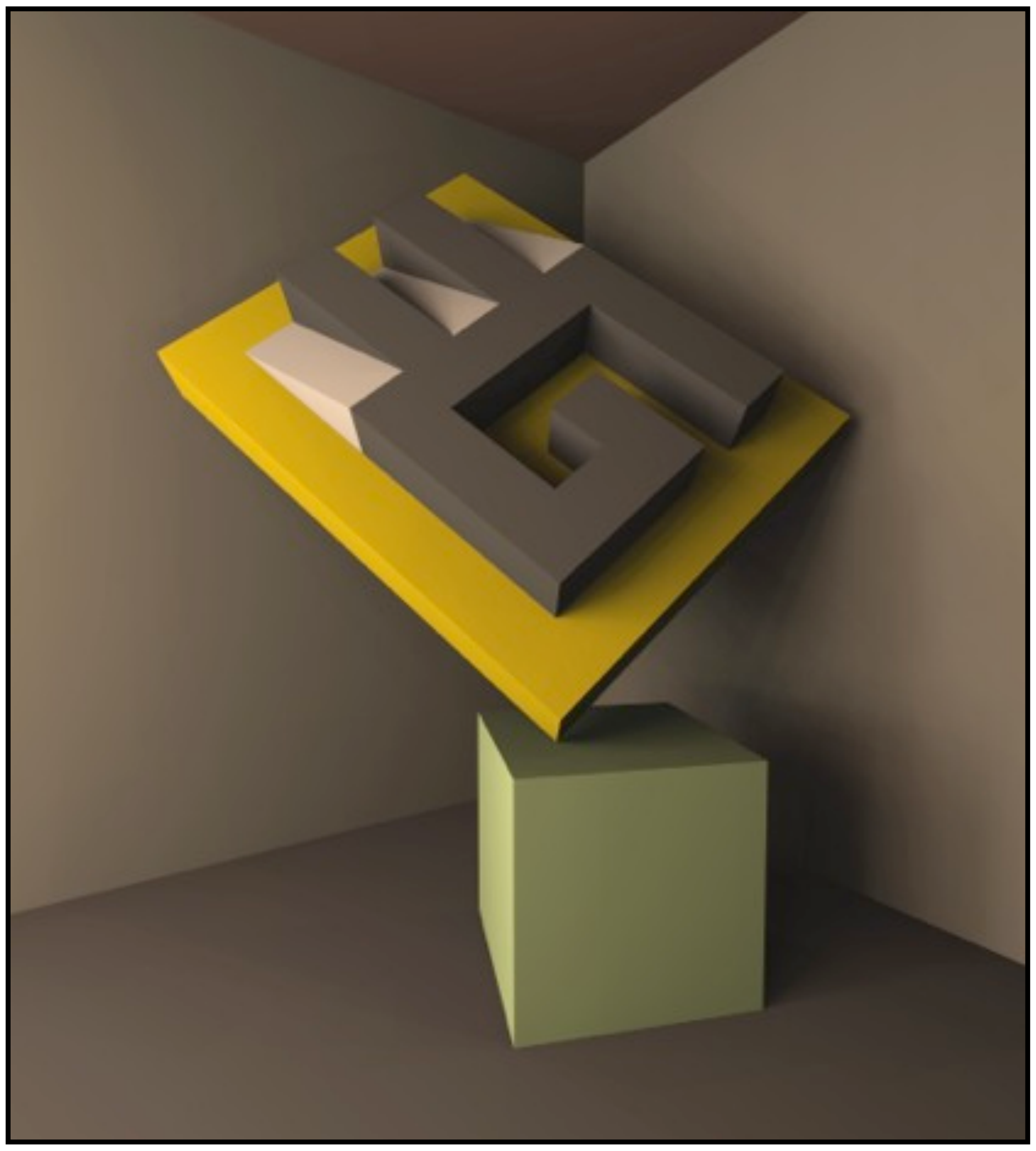
Improved Generation of VPLs

Rejection of unimportant VPLs [Georgiev and Slusallek 2010]

Without rejection



With rejection (7% acceptance)



Images courtesy of Georgiev and Slusallek

Here we have some example results that use the same number of VPLs to illuminate the scene. You can see that the rejection technique reduces the structured artifacts.

Improved Generation of VPLs

A more advanced technique builds on bidirectional sampling of light transport.

It was presented at the rendering workshop 2006 by Segovia and colleagues,

and the idea is to use bidirectional path tracing to construct a path and then place a VPL at the 2nd vertex from the camera.

This VPL will illuminate many points seen by the camera and thus the possibly expensive construction of the path is well amortized.

Improved Generation of VPLs

Bidirectional sampling for VPL distributions [Segovia et al. 2006]
aka “Bidirectional Instant Radiosity”

A more advanced technique builds on bidirectional sampling of light transport.

It was presented at the rendering workshop 2006 by Segovia and colleagues,

and the idea is to use bidirectional path tracing to construct a path and then place a VPL at the 2nd vertex from the camera.

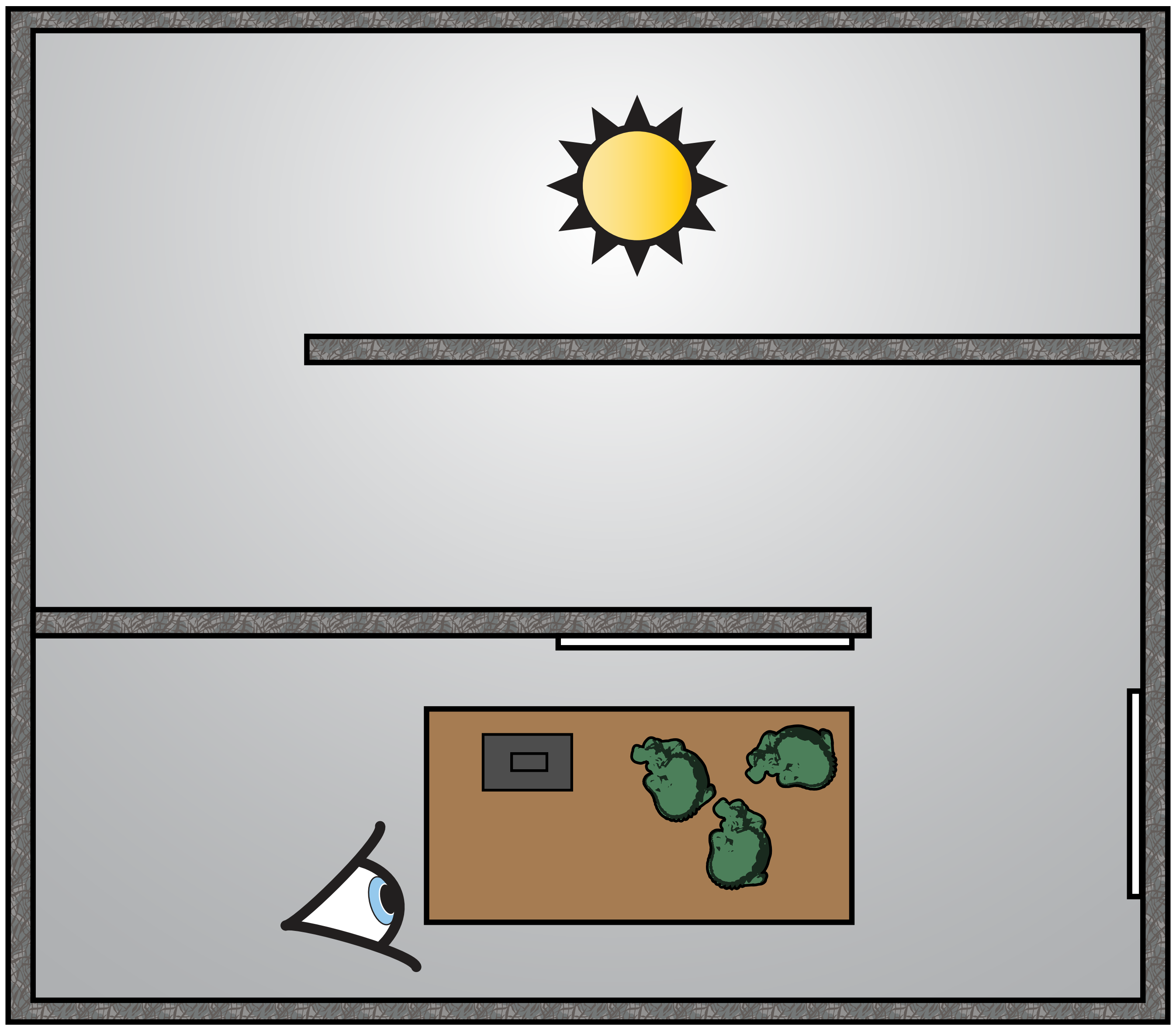
This VPL will illuminate many points seen by the camera and thus the possibly expensive construction of the path is well amortized.

Improved Generation of VPLs

Bidirectional sampling for VPL distributions [Segovia et al. 2006]
aka "Bidirectional Instant Radiosity"

Idea:

- create a VPL at the 2nd bounce from the camera



A more advanced technique builds on bidirectional sampling of light transport.

It was presented at the rendering workshop 2006 by Segovia and colleagues,

and the idea is to use bidirectional path tracing to construct a path and then place a VPL at the 2nd vertex from the camera.

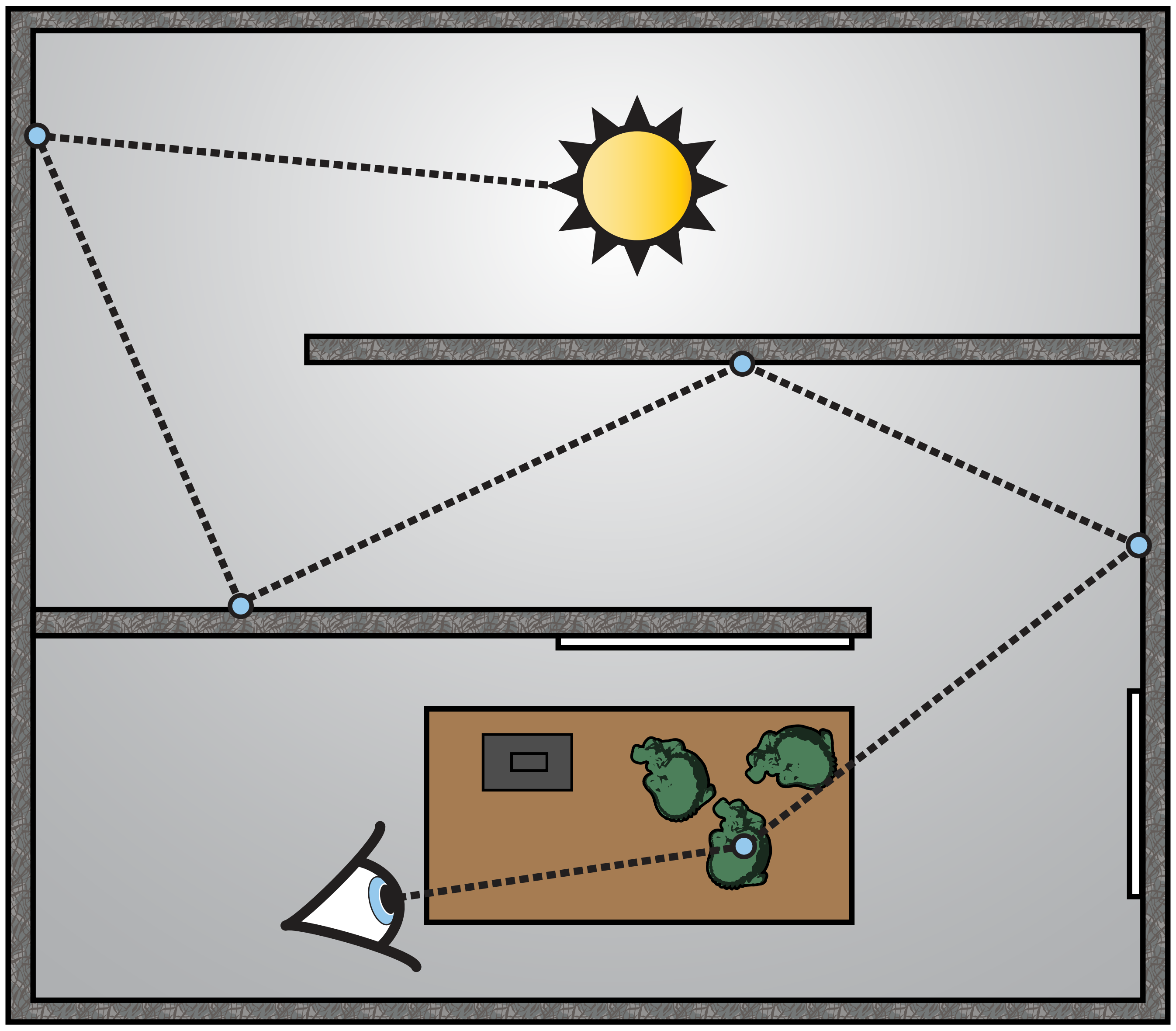
This VPL will illuminate many points seen by the camera and thus the possibly expensive construction of the path is well amortized.

Improved Generation of VPLs

Bidirectional sampling for VPL distributions [Segovia et al. 2006]
aka "Bidirectional Instant Radiosity"

Idea:

- create a VPL at the 2nd bounce from the camera



A more advanced technique builds on bidirectional sampling of light transport.

It was presented at the rendering workshop 2006 by Segovia and colleagues,

and the idea is to use bidirectional path tracing to construct a path and then place a VPL at the 2nd vertex from the camera.

This VPL will illuminate many points seen by the camera and thus the possibly expensive construction of the path is well amortized.

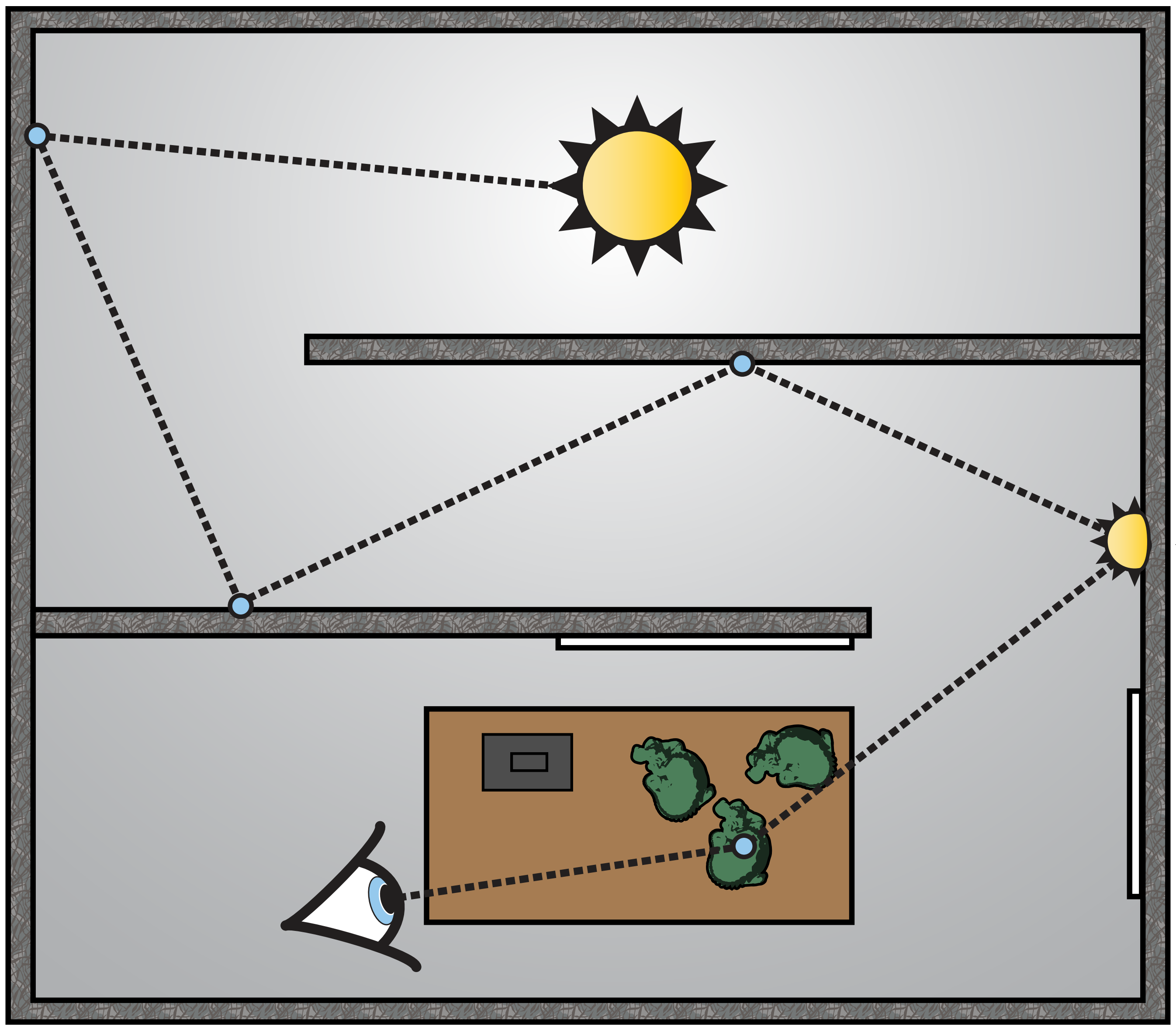
Improved Generation of VPLs

Bidirectional sampling for VPL distributions [Segovia et al. 2006]

aka “Bidirectional Instant Radiosity”

Idea:

- create a VPL at the 2nd bounce from the camera



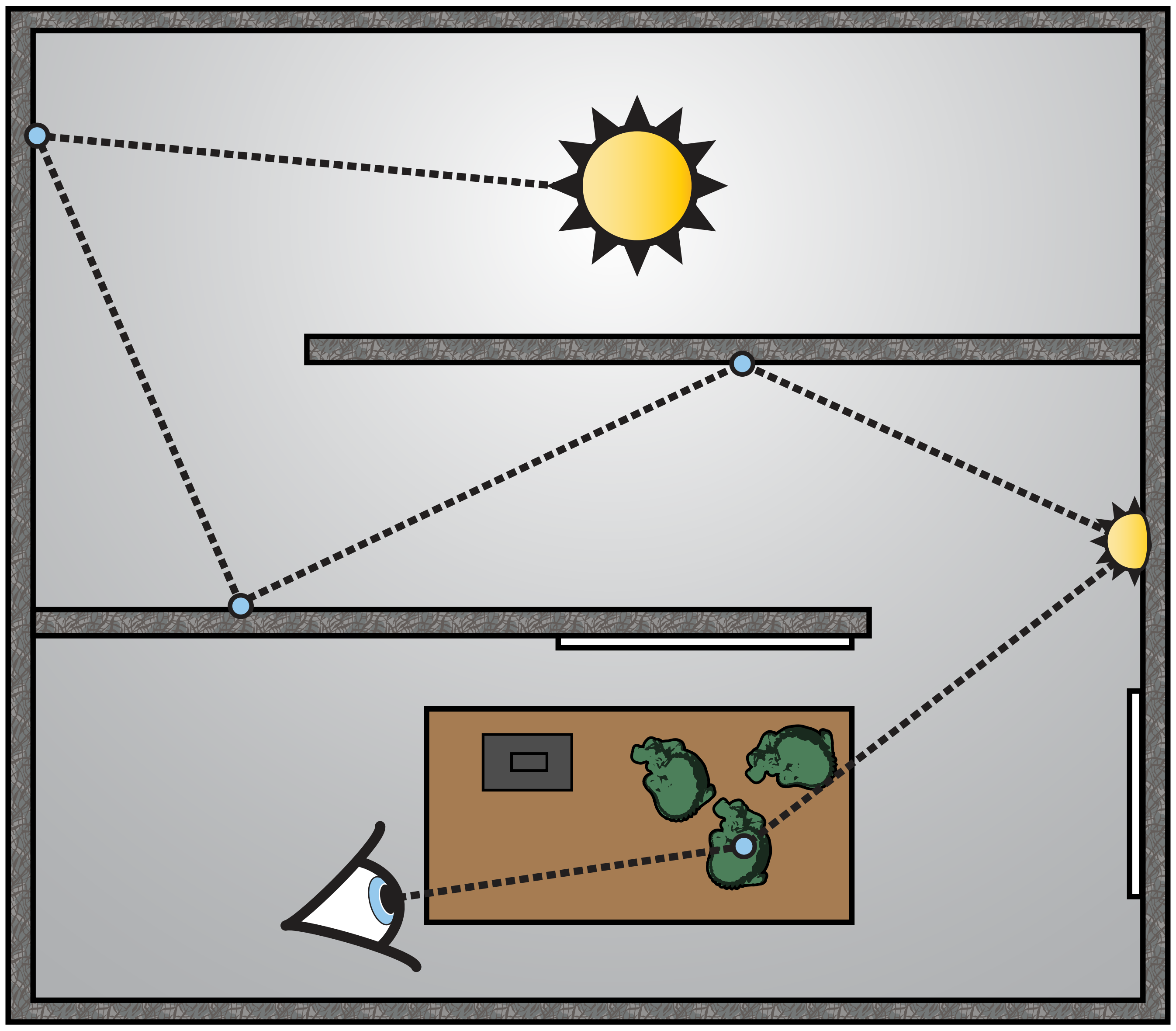
A more advanced technique builds on bidirectional sampling of light transport.

It was presented at the rendering workshop 2006 by Segovia and colleagues,

and the idea is to use bidirectional path tracing to construct a path and then place a VPL at the 2nd vertex from the camera.

This VPL will illuminate many points seen by the camera and thus the possibly expensive construction of the path is well amortized.

Improved Generation of VPLs

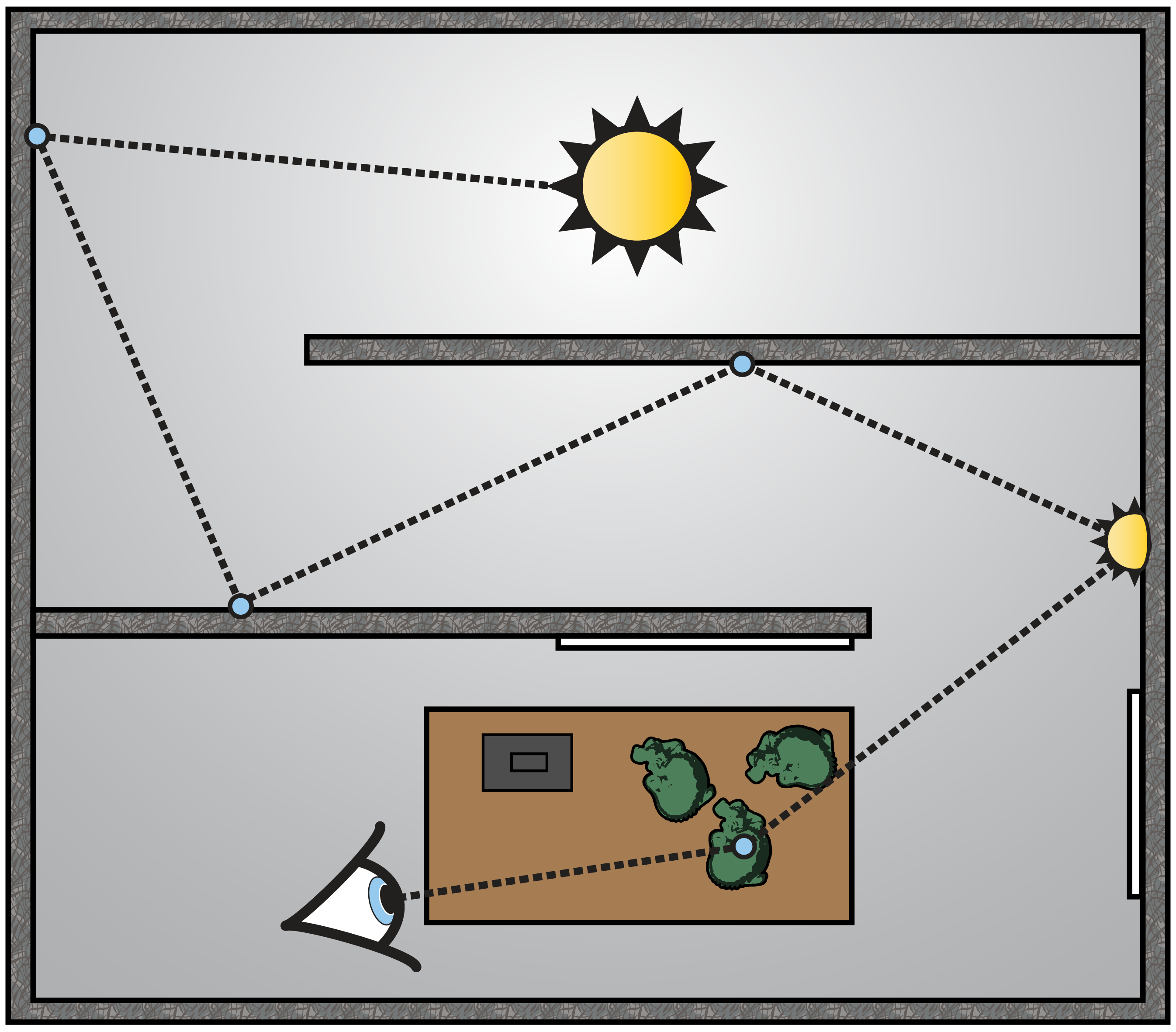


The same authors then extended the technique to generate more VPLs around the second vertex using Metropolis-Hastings sampling.

They mutate the paths to create VPLs that have an equal contribution to the image.

Improved Generation of VPLs

Metropolis sampling for VPL distributions [Segovia et al. 2007]
aka "Metropolis Instant Radiosity"



The same authors then extended the technique to generate more VPLs around the second vertex using Metropolis-Hastings sampling.

They mutate the paths to create VPLs that have an equal contribution to the image.

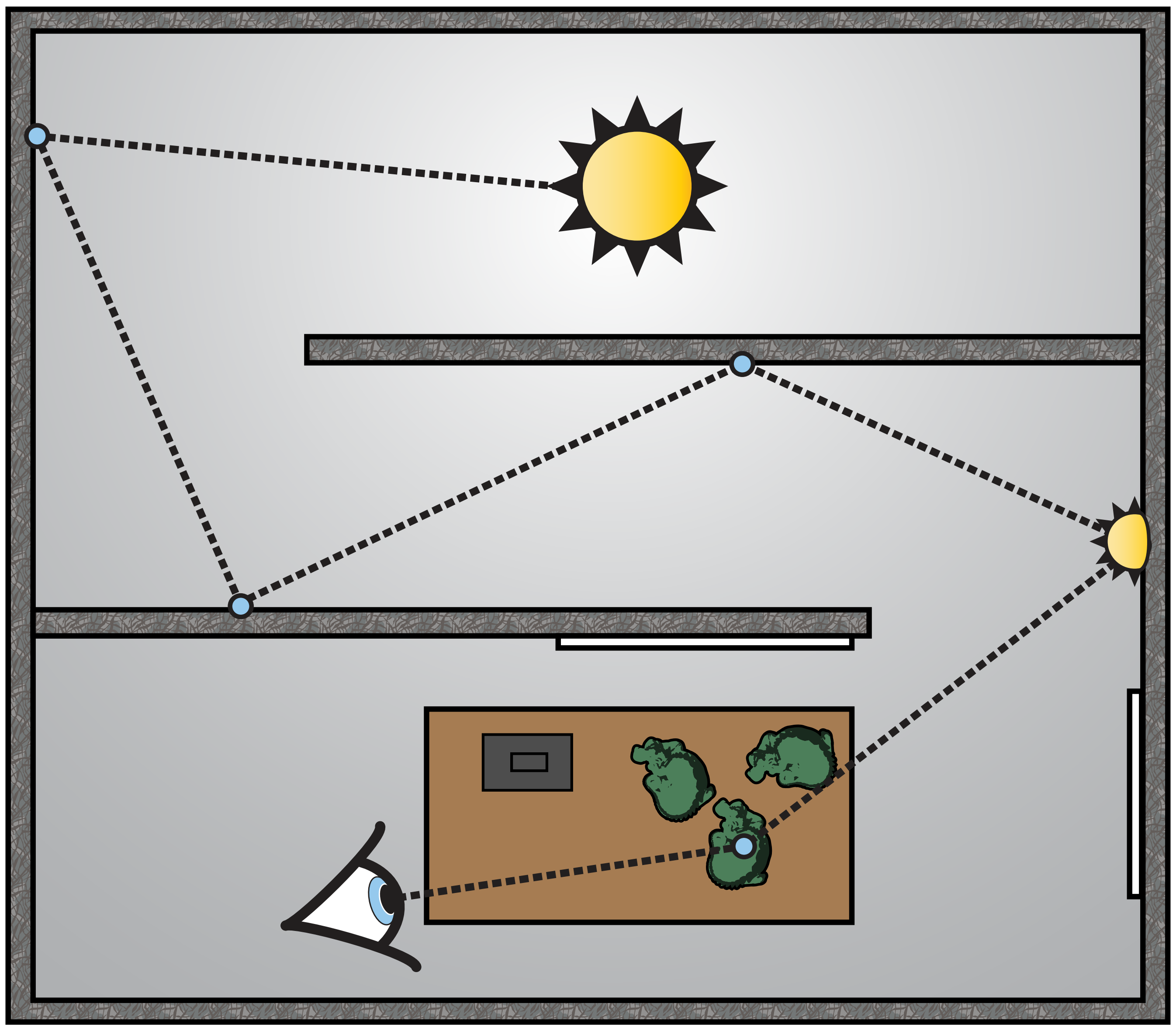
Improved Generation of VPLs

Metropolis sampling for VPL distributions [Segovia et al. 2007]

aka "Metropolis Instant Radiosity"

Idea:

- generate VPLs by mutating paths



The same authors then extended the technique to generate more VPLs around the second vertex using Metropolis-Hastings sampling.

They mutate the paths to create VPLs that have an equal contribution to the image.

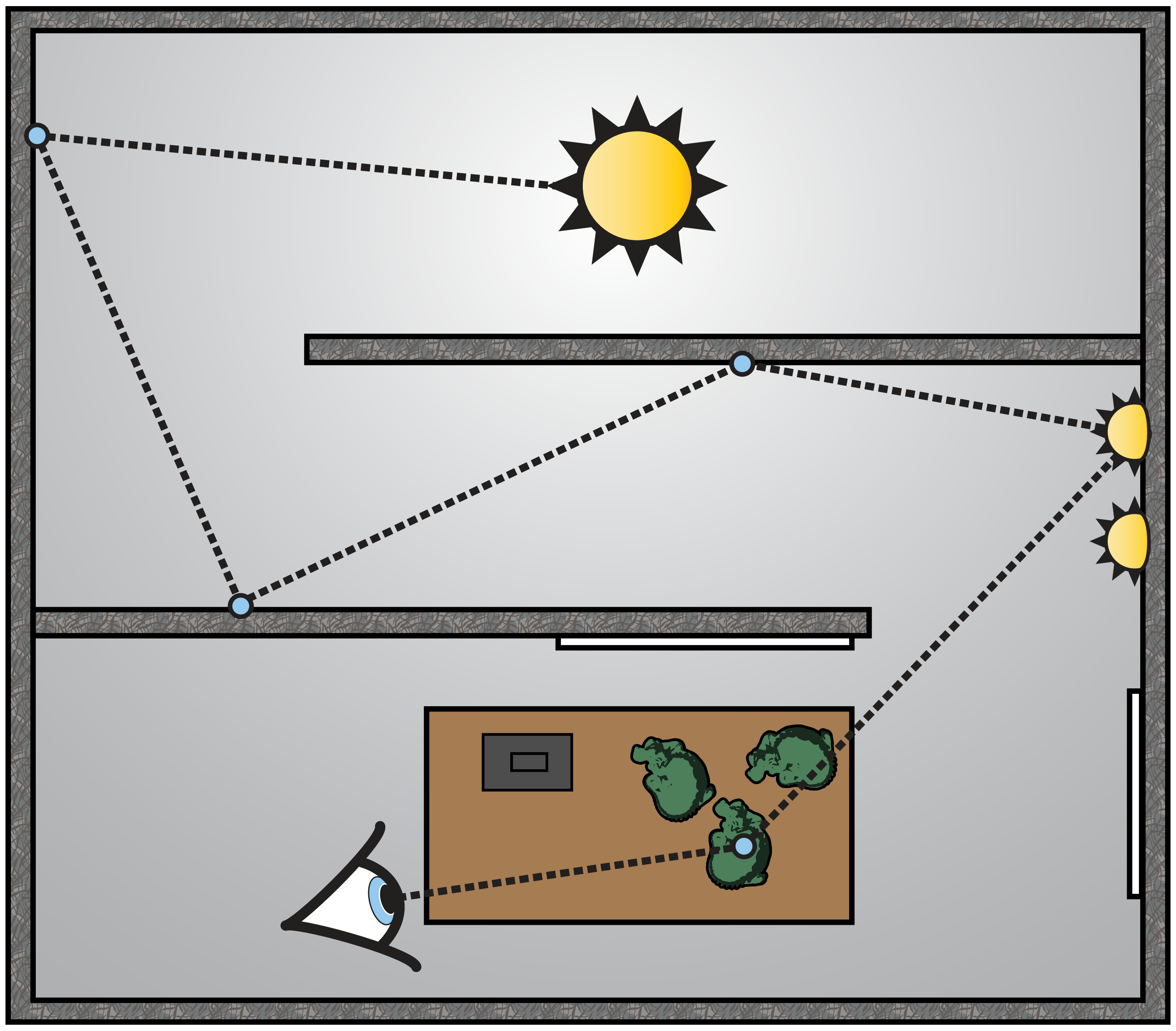
Improved Generation of VPLs

Metropolis sampling for VPL distributions [Segovia et al. 2007]

aka “Metropolis Instant Radiosity”

Idea:

- generate VPLs by mutating paths



The same authors then extended the technique to generate more VPLs around the second vertex using Metropolis-Hastings sampling.

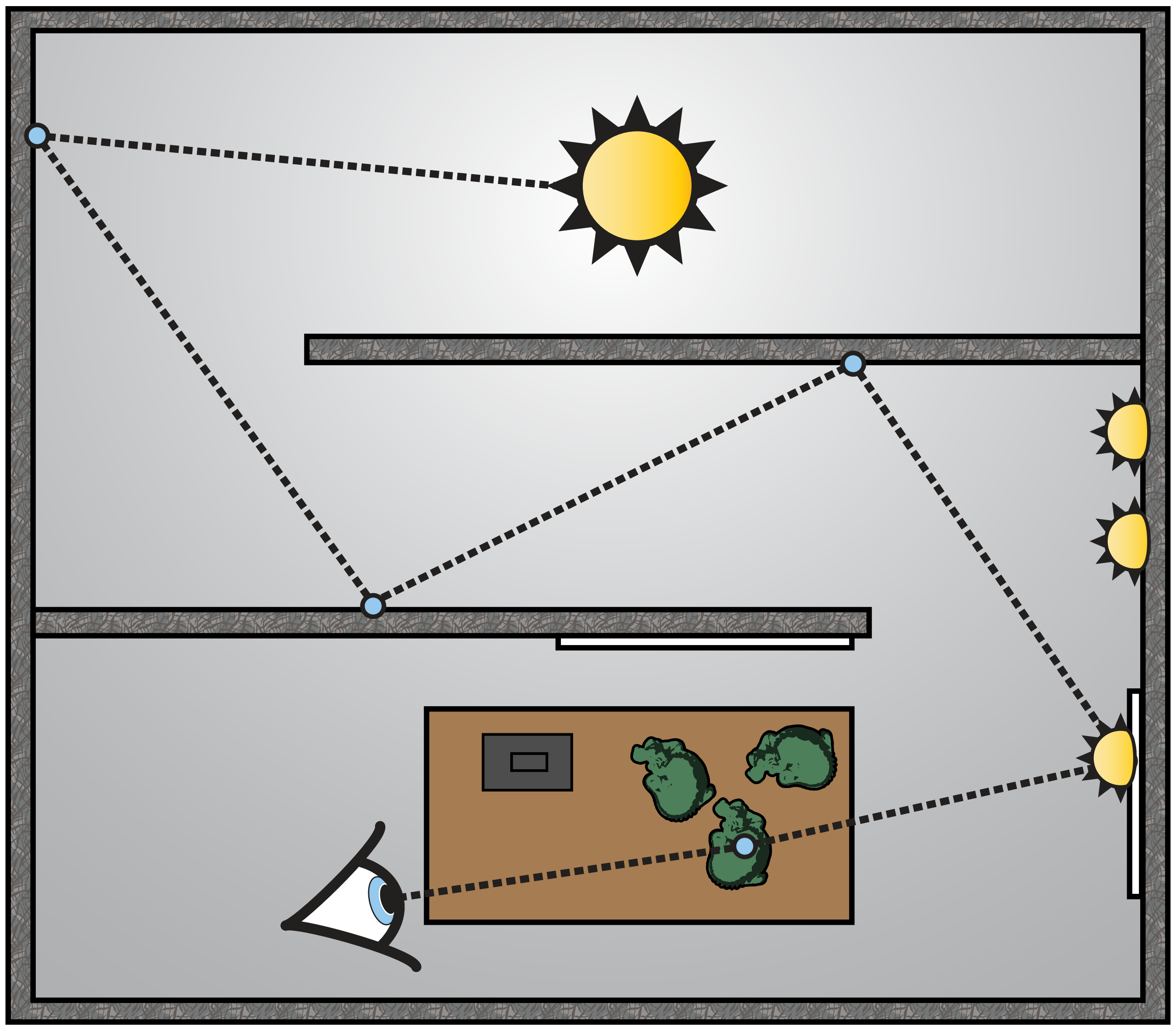
They mutate the paths to create VPLs that have an equal contribution to the image.

Improved Generation of VPLs

Metropolis sampling for VPL distributions [Segovia et al. 2007]
aka “Metropolis Instant Radiosity”

Idea:

- generate VPLs by mutating paths



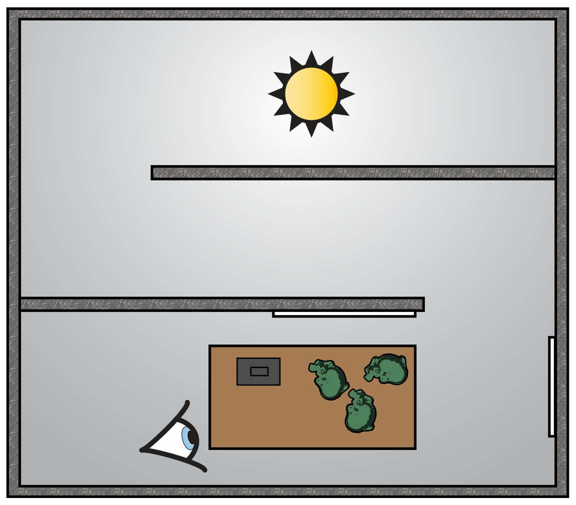
The same authors then extended the technique to generate more VPLs around the second vertex using Metropolis-Hastings sampling.

They mutate the paths to create VPLs that have an equal contribution to the image.

Improved Generation of VPLs

Metropolis sampling for VPL distributions [Segovia et al. 2007]
aka "Metropolis Instant Radiosity"

- comparison with equal number of VPLs (1024)



Here we see a comparison of instant radiosity with naive generation of VPLs,

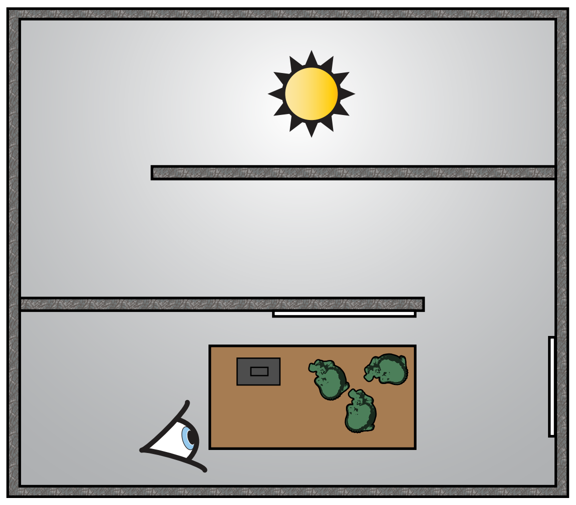
Bidirectional Instant radiosity (in the middle) and metropolis instant radiosity (on the right).

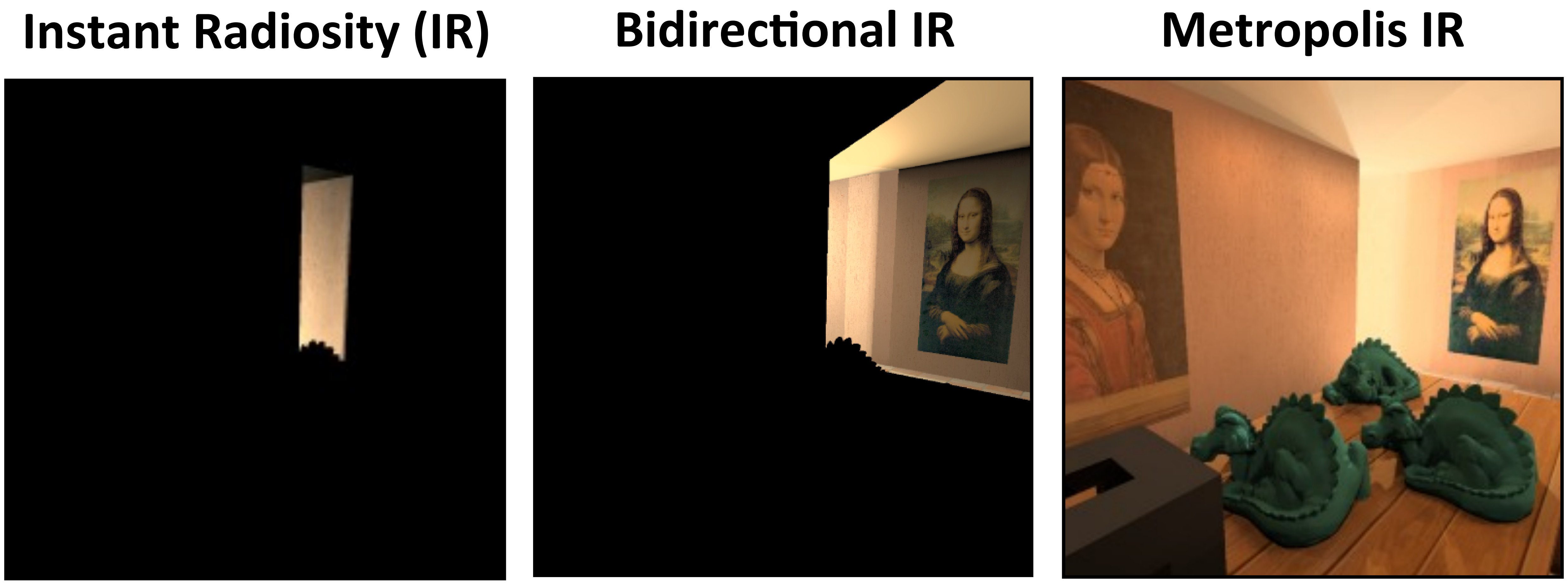
All of the images were computed using 1024 VPLs.

Improved Generation of VPLs

Metropolis sampling for VPL distributions [Segovia et al. 2007]
aka "Metropolis Instant Radiosity"

- comparison with equal number of VPLs (1024)





Images courtesy of Segovia et al.

Here we see a comparison of instant radiosity with naive generation of VPLs,

Bidirectional Instant radiosity (in the middle) and metropolis instant radiosity (on the right).

All of the images were computed using 1024 VPLs.

Improved Generation of VPLs

Metropolis sampling for VPL distributions [Segovia et al. 2007]

aka “Metropolis Instant Radiosity”

The main advantage of Metropolis instant radiosity is the robustness, allowing to handle very complex scenes.

Once the algorithm discovers a path that contributes to the image, it amortizes its construction by creating several VPLs with equal contribution, which is the main advantage over the Bidirectional instant radiosity.

The main drawback is the more involved implementation. Also, it does not help much with the problem if glossy inter-reflections.

Improved Generation of VPLs

Metropolis sampling for VPL distributions [Segovia et al. 2007]
aka "Metropolis Instant Radiosity"

Advantages:

- handles large and difficult scenes
- VPLs have equal contribution

The main advantage of Metropolis instant radiosity is the robustness, allowing to handle very complex scenes.

Once the algorithm discovers a path that contributes to the image, it amortizes its construction by creating several VPLs with equal contribution, which is the main advantage over the Bidirectional instant radiosity.

The main drawback is the more involved implementation. Also, it does not help much with the problem if glossy inter-reflections.

Improved Generation of VPLs

Metropolis sampling for VPL distributions [Segovia et al. 2007]
aka "Metropolis Instant Radiosity"

Advantages:

- handles large and difficult scenes
- VPLs have equal contribution

Disadvantages:

- complicated implementation
- does not help with local inter-reflections

The main advantage of Metropolis instant radiosity is the robustness, allowing to handle very complex scenes.

Once the algorithm discovers a path that contributes to the image, it amortizes its construction by creating several VPLs with equal contribution, which is the main advantage over the Bidirectional instant radiosity.

The main drawback is the more involved implementation. Also, it does not help much with the problem if glossy inter-reflections.

Improved Generation of VPLs

The last technique that we will briefly mention tries to address this problem of glossy inter-reflection.

And it does so by creating a set of global VPLs from the light sources and a set of local lights from the camera.

We will detail this technique later in the bias compensation section, where it fits better context-wise.

Improved Generation of VPLs

Local Virtual Lights [Davidovič et al. 2010]

- improves glossy inter-reflections
- split light transport into a **global** and **local** component

The last technique that we will briefly mention tries to address this problem of glossy inter-reflection.

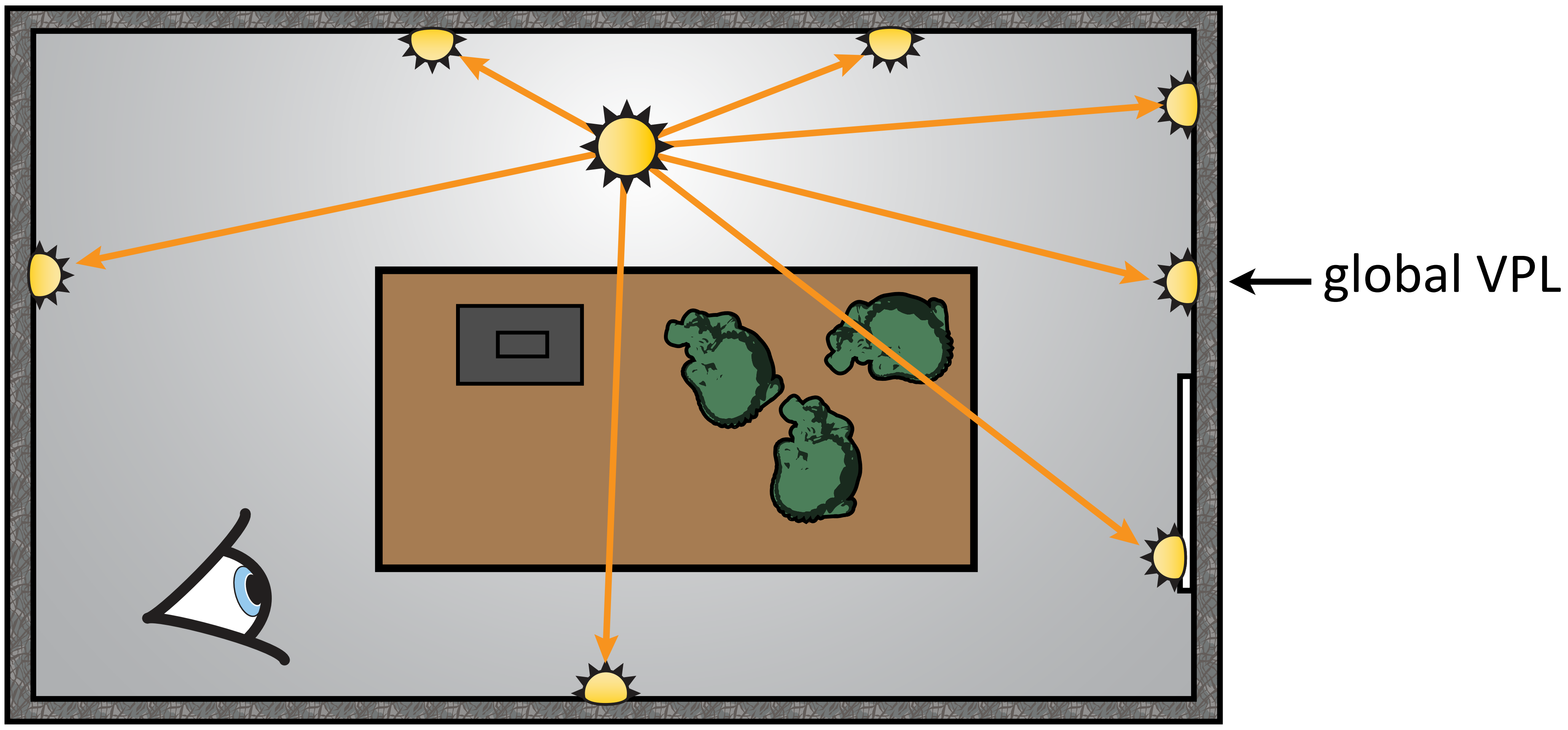
And it does so by creating a set of global VPLs from the light sources and a set of local lights from the camera.

We will detail this technique later in the bias compensation section, where it fits better context-wise.

Improved Generation of VPLs

Local Virtual Lights [Davidovič et al. 2010]

- improves glossy inter-reflections
- split light transport into a **global** and **local** component



The last technique that we will briefly mention tries to address this problem of glossy inter-reflection.

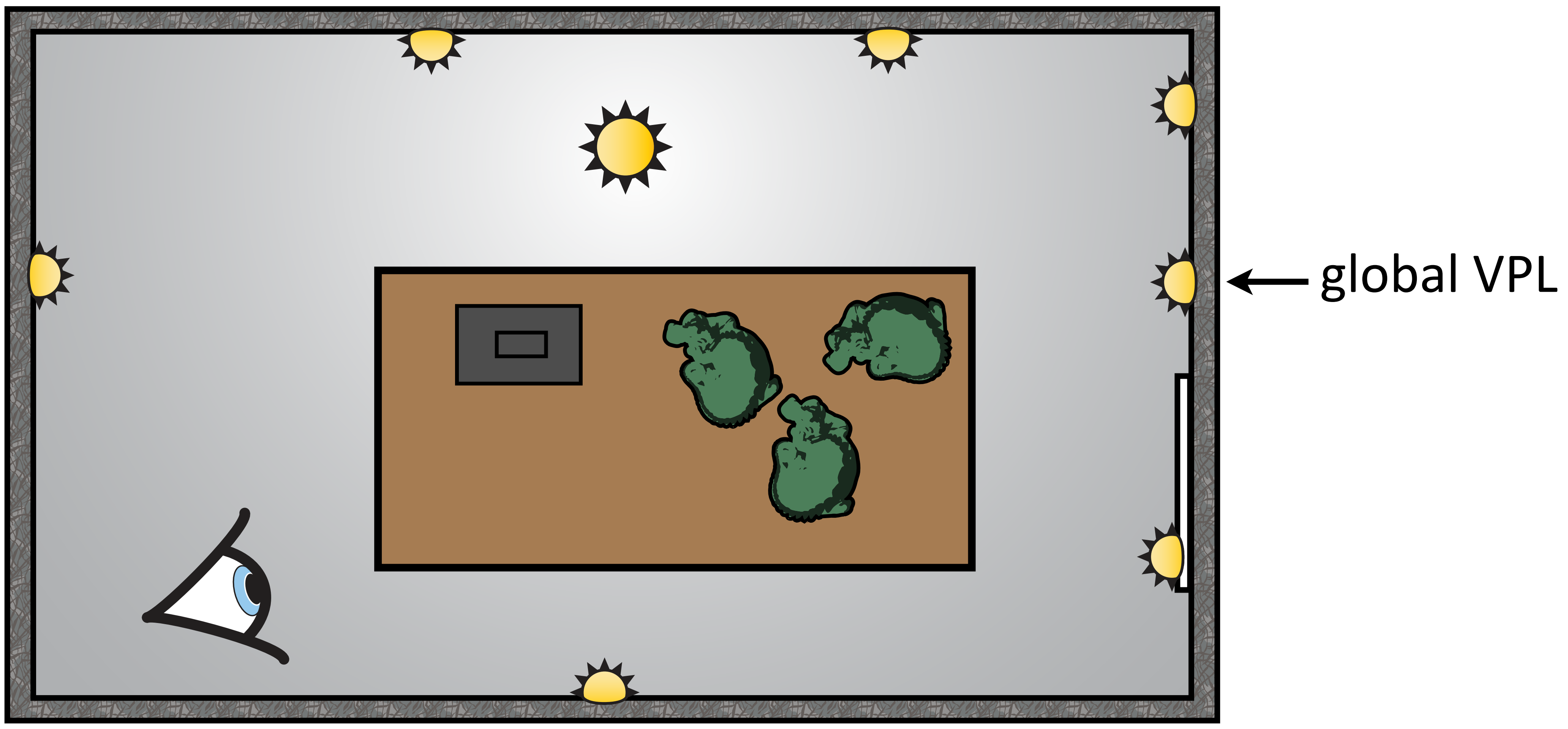
And it does so by creating a set of global VPLs from the light sources and a set of local lights from the camera.

We will detail this technique later in the bias compensation section, where it fits better context-wise.

Improved Generation of VPLs

Local Virtual Lights [Davidovič et al. 2010]

- improves glossy inter-reflections
- split light transport into a **global** and **local** component



The last technique that we will briefly mention tries to address this problem of glossy inter-reflection.

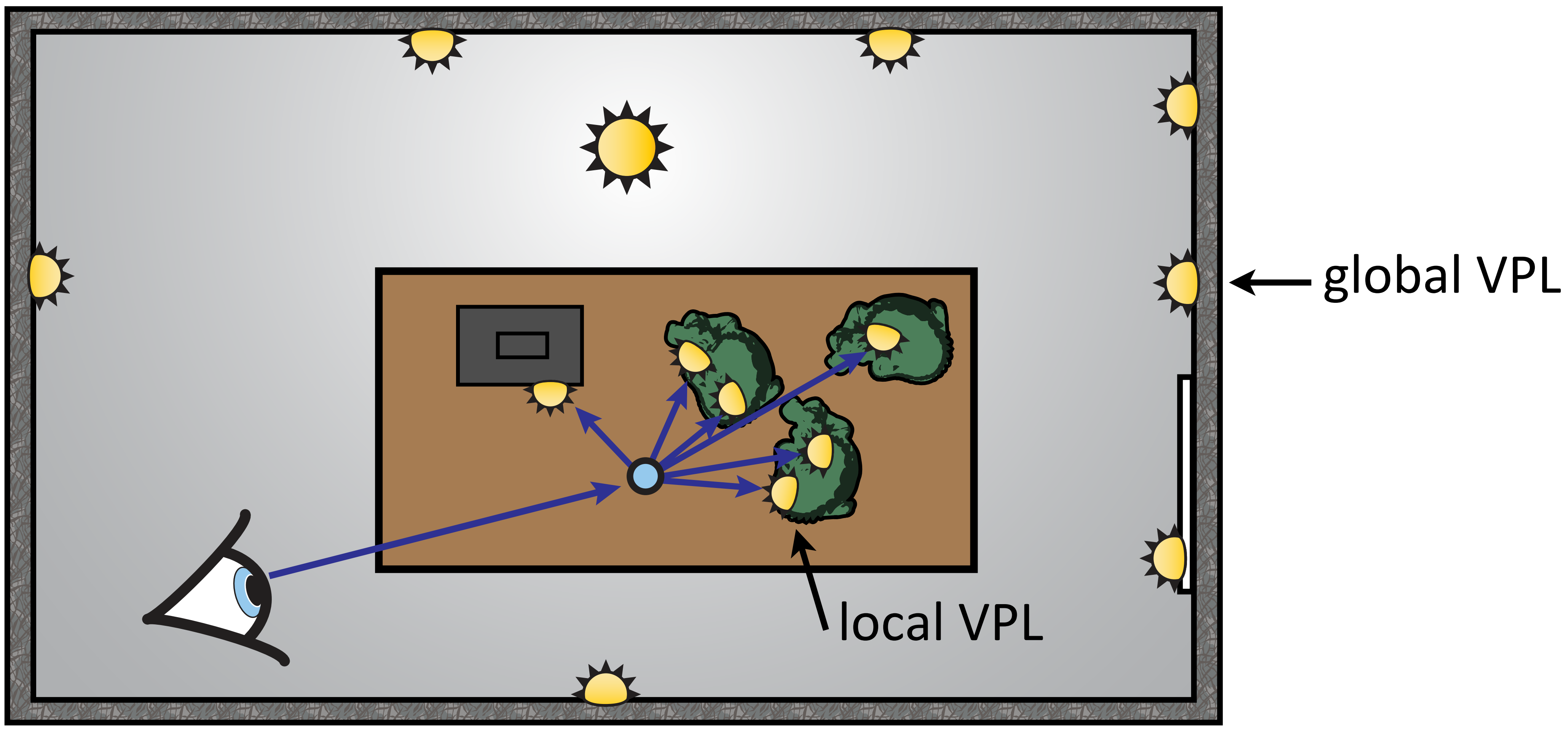
And it does so by creating a set of global VPLs from the light sources and a set of local lights from the camera.

We will detail this technique later in the bias compensation section, where it fits better context-wise.

Improved Generation of VPLs

Local Virtual Lights [Davidovič et al. 2010]

- improves glossy inter-reflections
- split light transport into a **global** and **local** component
- details are mentioned later



The last technique that we will briefly mention tries to address this problem of glossy inter-reflection.

And it does so by creating a set of global VPLs from the light sources and a set of local lights from the camera.

We will detail this technique later in the bias compensation section, where it fits better context-wise.

Improved Generation of VPLs

Comparison

Here we have the three most interesting techniques compared in a table.

The rejection of VPLs is very simple to implement and works reasonably well in moderately complex scenes.

The metropolis light transport is capable of handling pretty much all types of geometry, but at a significant implementation cost.

None of these two techniques explicitly addresses the problem of glossy inter-reflections, that’s where the local lights can help a lot.

Improved Generation of VPLs

Comparison

	Moderately complex scenes	Complex scenes	Glossy scenes	Implementation
Rejection of VPLs [Georgiev and Slusallek 2010]	✓	✗	✗	✓
Metropolis IR [Segovia et al. 2007]	✓	✓	✗	✗
Local Virtual Lights [Davidovič et al. 2010]	✓	✗	✓	✗

✓	yes, easy
✗	no, difficult

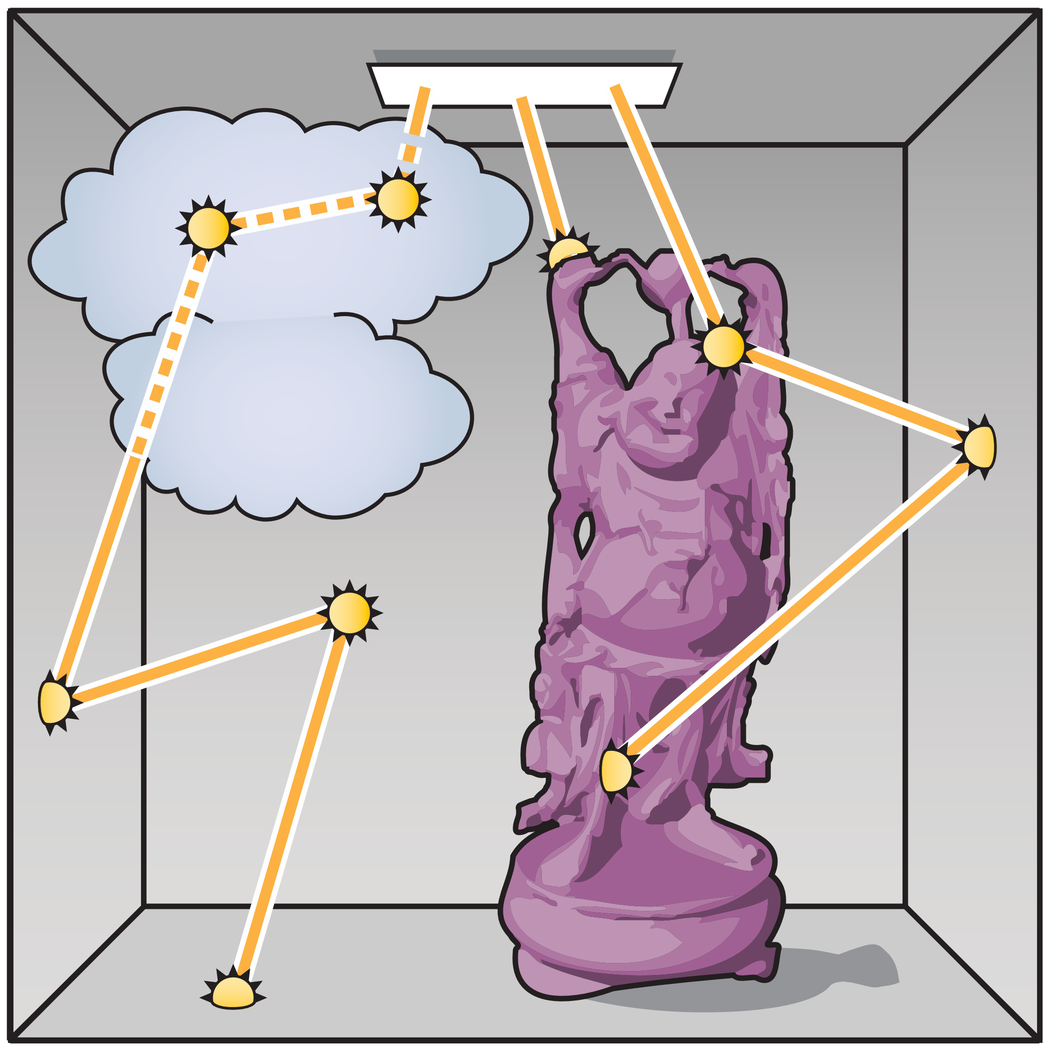
Here we have the three most interesting techniques compared in a table.

The rejection of VPLs is very simple to implement and works reasonably well in moderately complex scenes.

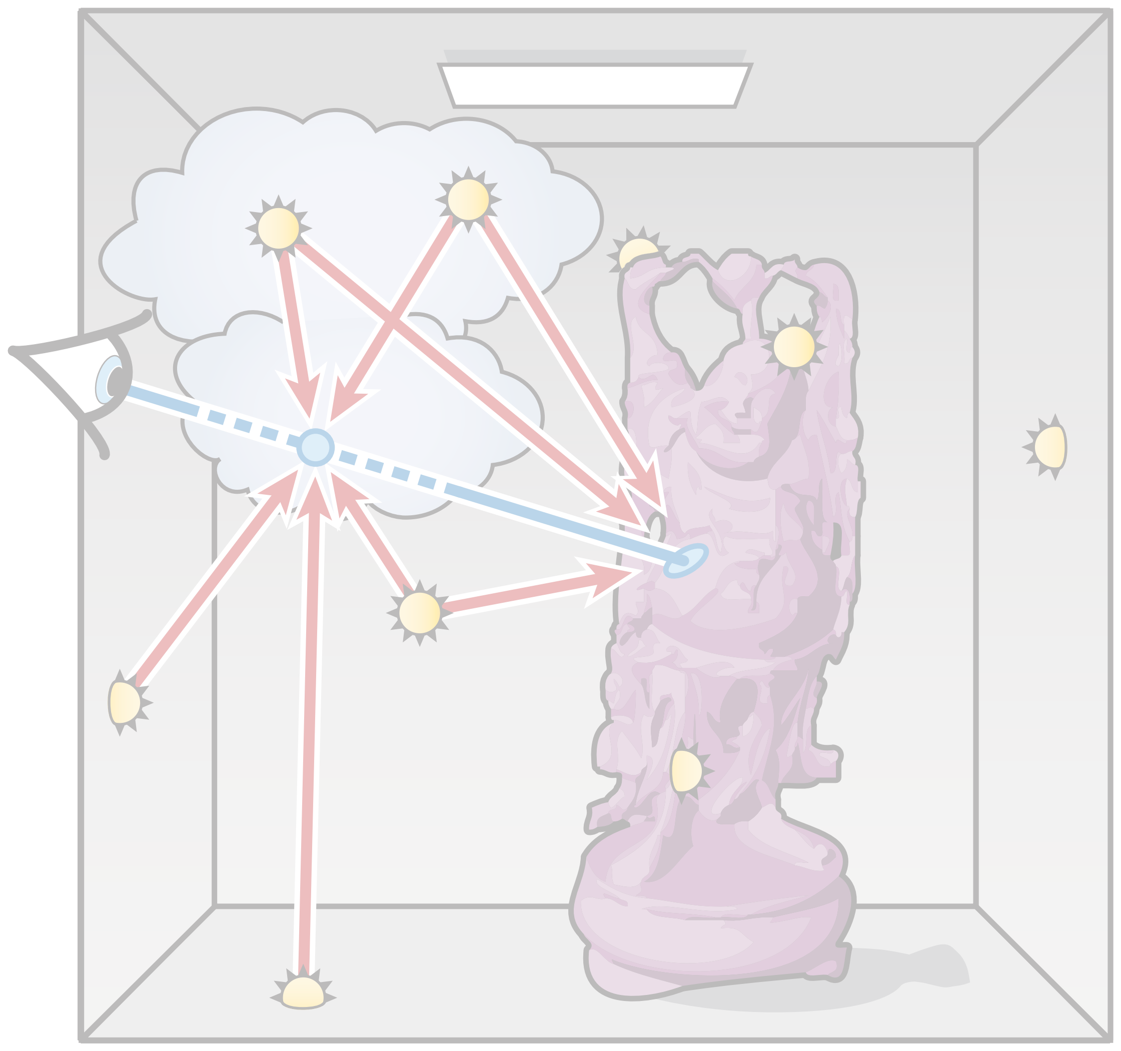
The metropolis light transport is capable of handling pretty much all types of geometry, but at a significant implementation cost.

None of these two techniques explicitly addresses the problem of glossy inter-reflections, that's where the local lights can help a lot.

Many-light Methods



Generation of VPLs

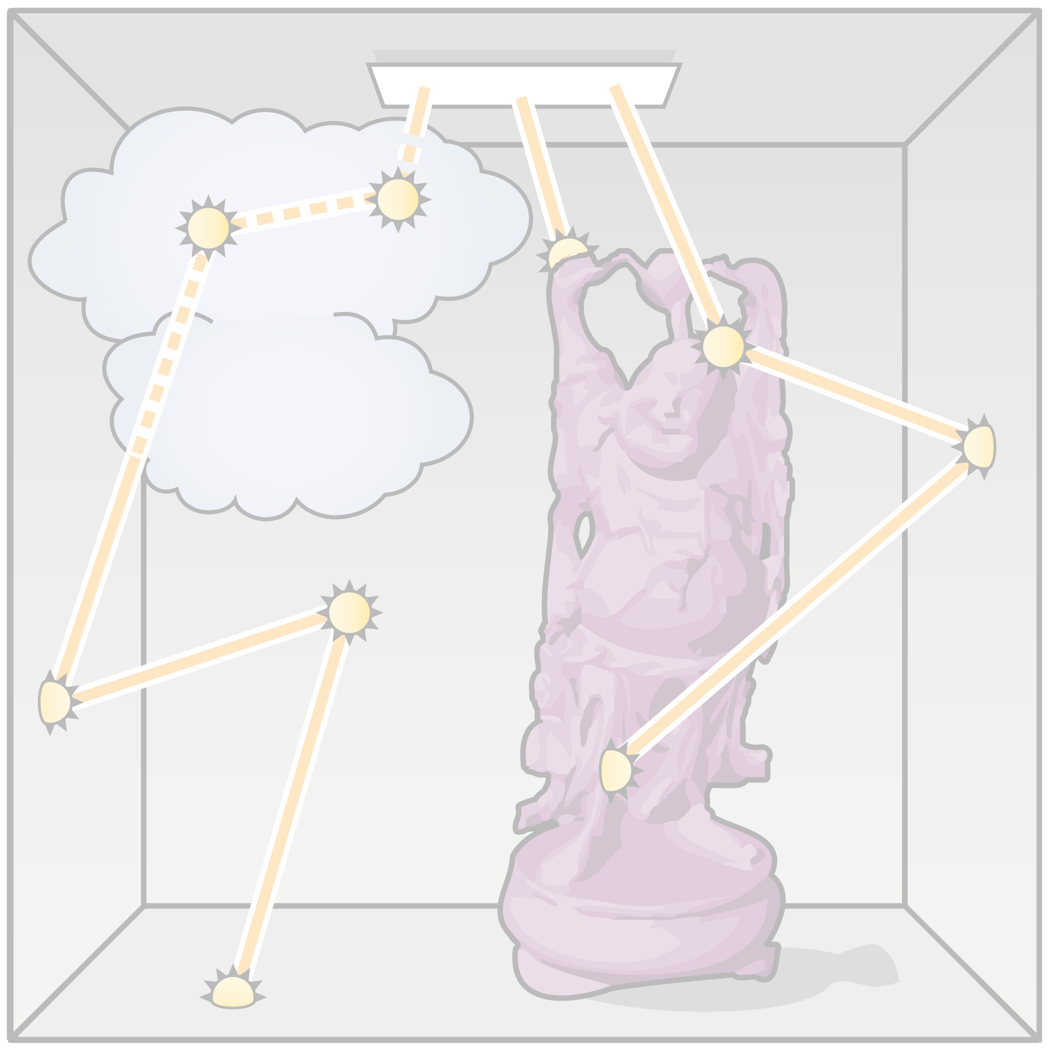


Lighting with VPLs

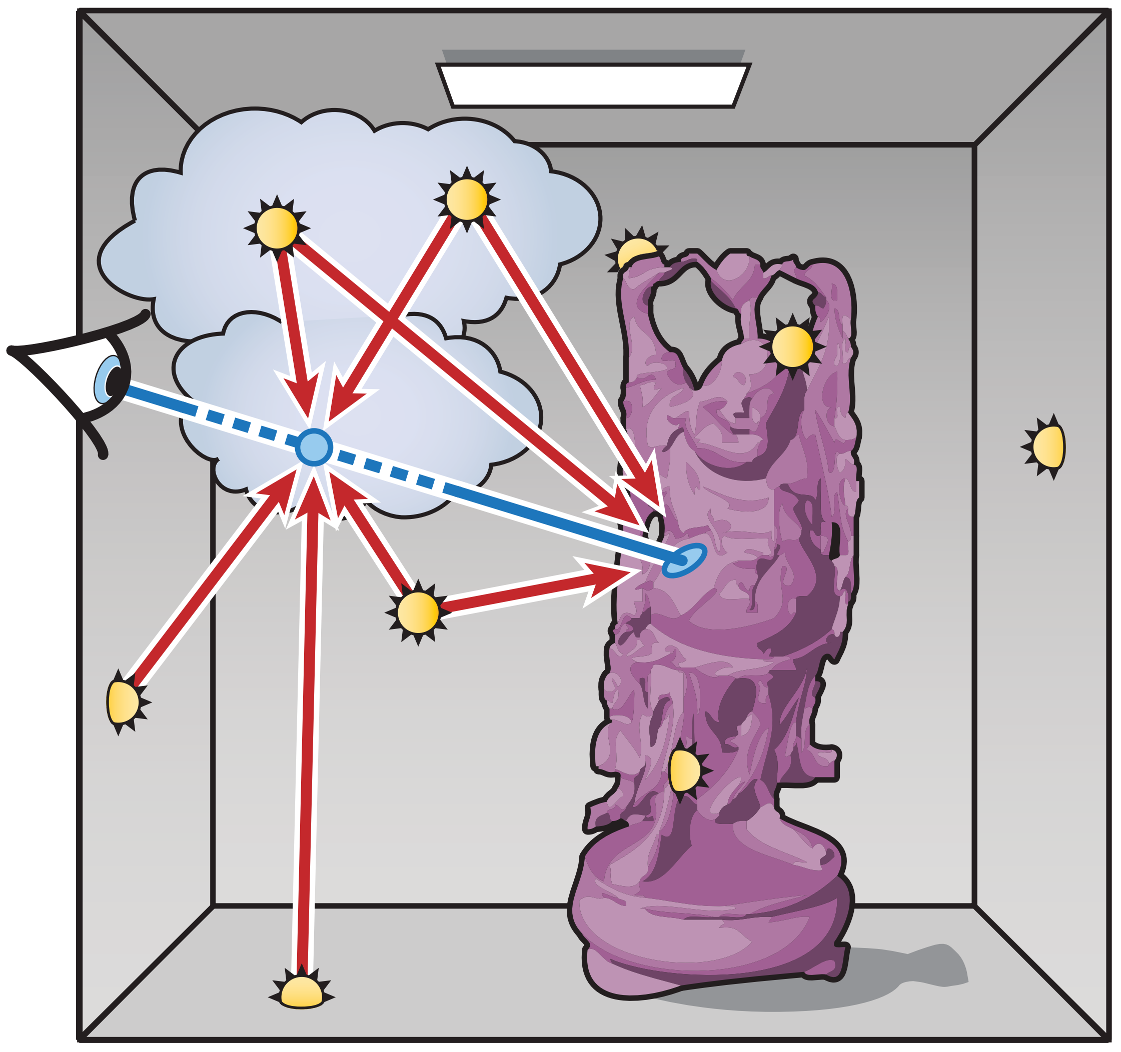
So far we have talked about the generation of VPLs.

Now we should have a detail look at how we use them to illuminate the scene.

Many-light Methods



Generation of VPLs



Lighting with VPLs

So far we have talked about the generation of VPLs.

Now we should have a detail look at how we use them to illuminate the scene.

Lighting with VPLs (assuming no media)

We will consider the case when the scene does not contain any participating media.
The extensions necessary to handle the media can be found in the paper.

Here we have the rendering equation, let’s just very quickly revisit the individual terms:

f is the BRDF, G is the geometry term, which accounts for the mutual orientation and distance between the two points, V is the visibility between the points, and L is the outgoing radiance.

And you can see that the equation is recursive...

Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1 \to \mathbf{x}_0) \; = \; L_\mathrm{e}(\mathbf{x}_1 \to \mathbf{x}_0) + \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2)\, V(\mathbf{x}_1, \mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

We will consider the case when the scene does not contain any participating media.
The extensions necessary to handle the media can be found in the paper.

Here we have the rendering equation, let's just very quickly revisit the individual terms:

f is the BRDF, G is the geometry term, which accounts for the mutual orientation and distance between the two points, V is the visibility between the points, and L is the outgoing radiance.

And you can see that the equation is recursive...

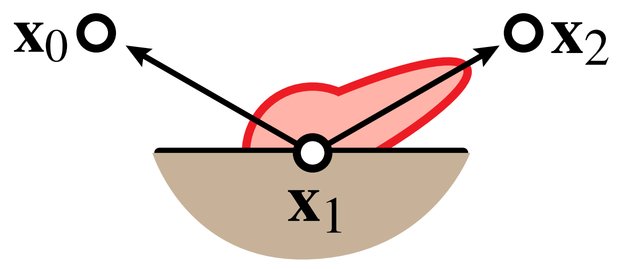
Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \int_A \boxed{f(\mathbf{x}_1)}\, G(\mathbf{x}_1, \mathbf{x}_2)\, V(\mathbf{x}_1, \mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

BRDF

$f(\mathbf{x}_1)$



We will consider the case when the scene does not contain any participating media. The extensions necessary to handle the media can be found in the paper.

Here we have the rendering equation, let's just very quickly revisit the individual terms:

f is the BRDF, G is the geometry term, which accounts for the mutual orientation and distance between the two points, V is the visibility between the points, and L is the outgoing radiance.

And you can see that the equation is recursive...

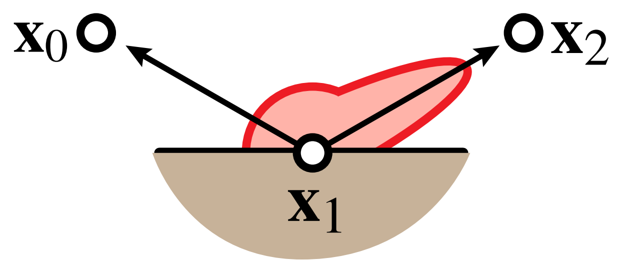
Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1\to\mathbf{x}_0) = L_e(\mathbf{x}_1\to\mathbf{x}_0) + \int_A f(\mathbf{x}_1)\, \boxed{G(\mathbf{x}_1,\mathbf{x}_2)}\, V(\mathbf{x}_1,\mathbf{x}_2)\, L(\mathbf{x}_2\to\mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

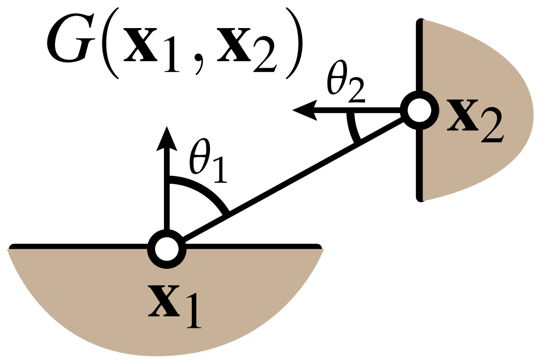
BRDF

$f(\mathbf{x}_1)$



Geometry term

$G(\mathbf{x}_1,\mathbf{x}_2)$



$$G(\mathbf{x}_1,\mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

We will consider the case when the scene does not contain any participating media.
The extensions necessary to handle the media can be found in the paper.

Here we have the rendering equation, let's just very quickly revisit the individual terms:

f is the BRDF, G is the geometry term, which accounts for the mutual orientation and distance between the two points, V is the visibility between the points, and L is the outgoing radiance.

And you can see that the equation is recursive...

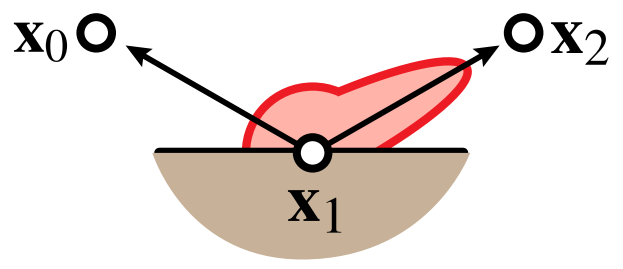
Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2)\, \boxed{V(\mathbf{x}_1, \mathbf{x}_2)}\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

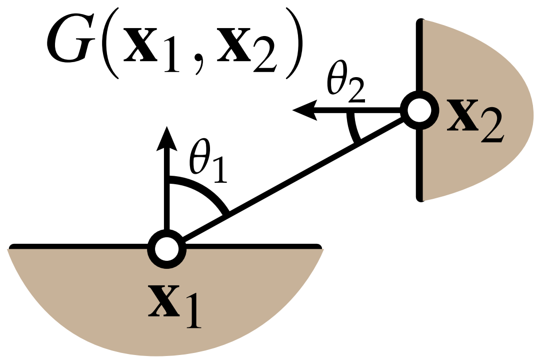
BRDF

$f(\mathbf{x}_1)$



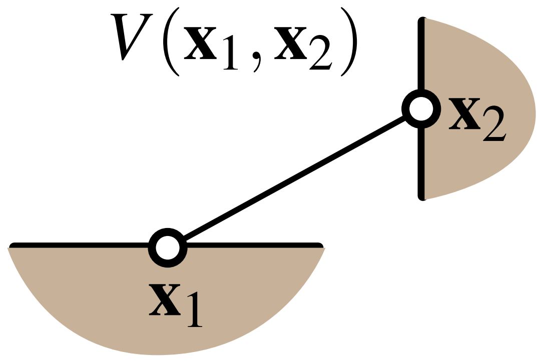
Geometry term

$G(\mathbf{x}_1, \mathbf{x}_2)$



Visibility term

$V(\mathbf{x}_1, \mathbf{x}_2)$



$$G(\mathbf{x}_1, \mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

We will consider the case when the scene does not contain any participating media.
The extensions necessary to handle the media can be found in the paper.

Here we have the rendering equation, let's just very quickly revisit the individual terms:

f is the BRDF, G is the geometry term, which accounts for the mutual orientation and distance between the two points, V is the visibility between the points, and L is the outgoing radiance.

And you can see that the equation is recursive...

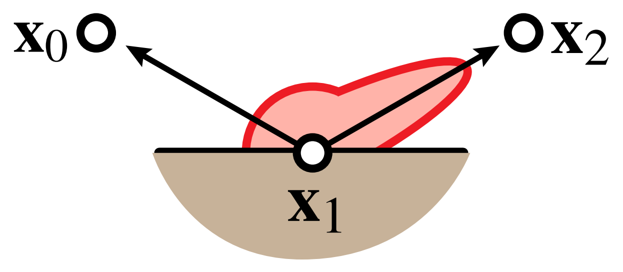
Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$\boxed{L(\mathbf{x}_1 \to \mathbf{x}_0)} = L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2)\, V(\mathbf{x}_1, \mathbf{x}_2) \boxed{L(\mathbf{x}_2 \to \mathbf{x}_1)}\, \mathrm{d}A(\mathbf{x}_2)$$

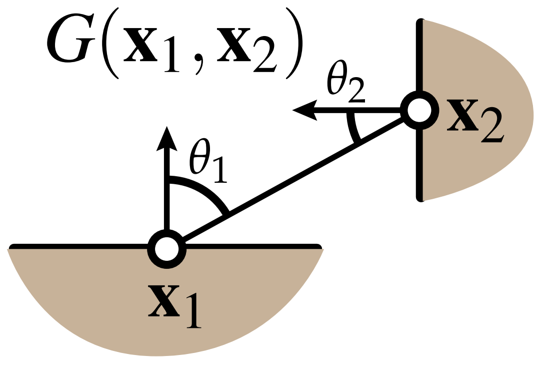
BRDF

$f(\mathbf{x}_1)$



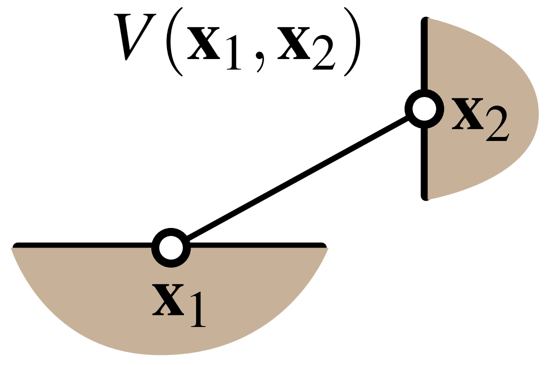
Geometry term

$G(\mathbf{x}_1, \mathbf{x}_2)$



Visibility term

$V(\mathbf{x}_1, \mathbf{x}_2)$



$$G(\mathbf{x}_1, \mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

We will consider the case when the scene does not contain any participating media. The extensions necessary to handle the media can be found in the paper.

Here we have the rendering equation, let’s just very quickly revisit the individual terms:

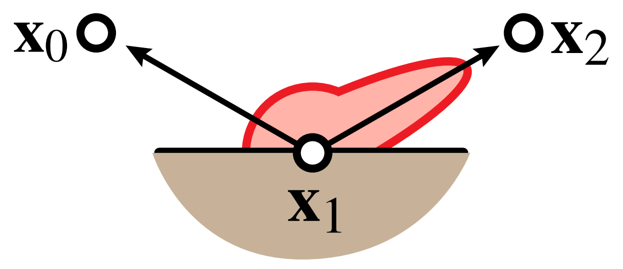
f is the BRDF, G is the geometry term, which accounts for the mutual orientation and distance between the two points, V is the visibility between the points, and L is the outgoing radiance.

And you can see that the equation is recursive...

Lighting with VPLs (assuming no media)

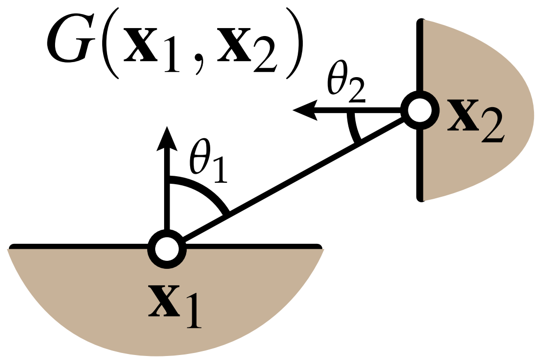
BRDF

$f(\mathbf{x}_1)$



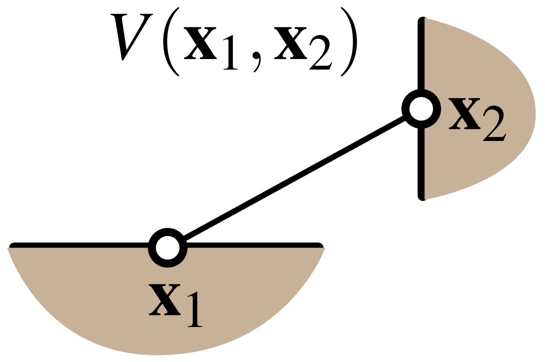
Geometry term

$G(\mathbf{x}_1, \mathbf{x}_2)$



Visibility term

$V(\mathbf{x}_1, \mathbf{x}_2)$



$$G(\mathbf{x}_1, \mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

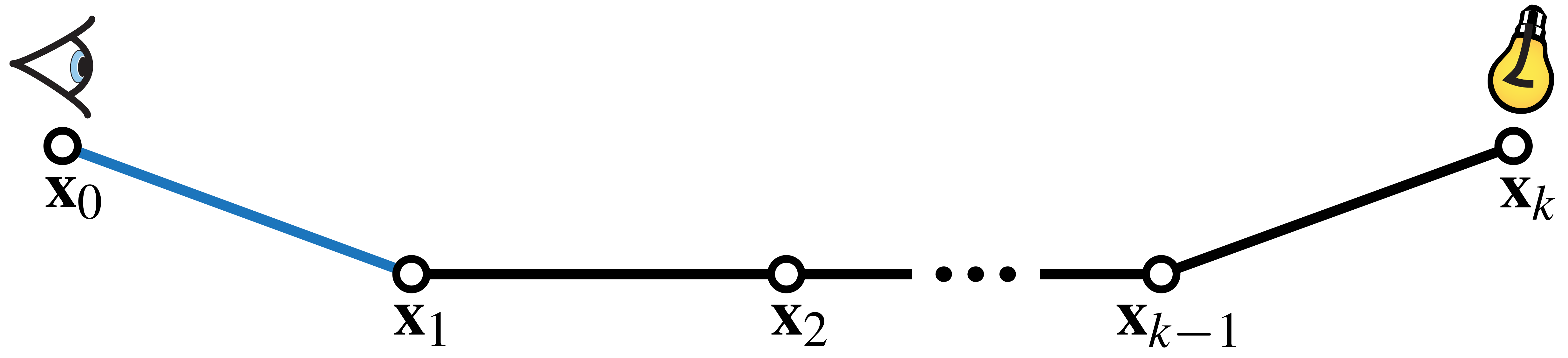
In order to solve the equation, we can construct paths between the camera and the light sources...

here you see all the terms defining the throughput of the path and as we mentioned previously, the main idea of instant radiosity is to collapse ... this ... subpath into a virtual light source.

And once we have these virtual lights, we just need to connect the shaded points to them and evaluate the four terms defined with respect to the red connection.

Lighting with VPLs (assuming no media)

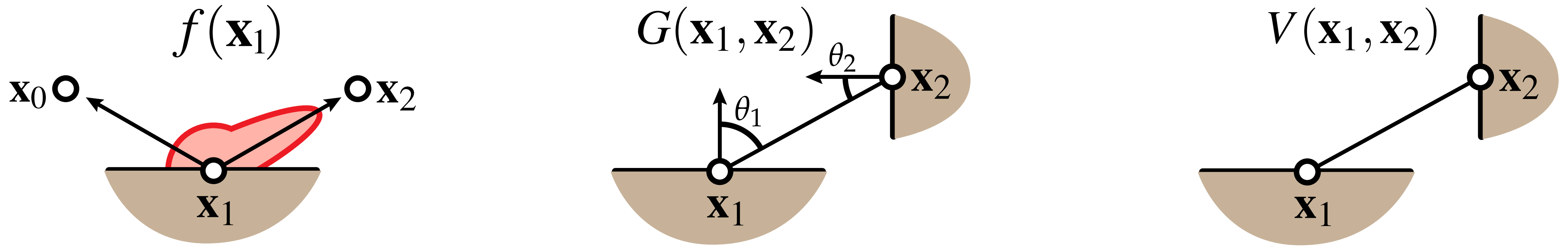
Paths between camera and light sources



BRDF $f(\mathbf{x}_1)$

Geometry term $G(\mathbf{x}_1, \mathbf{x}_2)$

Visibility term $V(\mathbf{x}_1, \mathbf{x}_2)$



$$G(\mathbf{x}_1, \mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

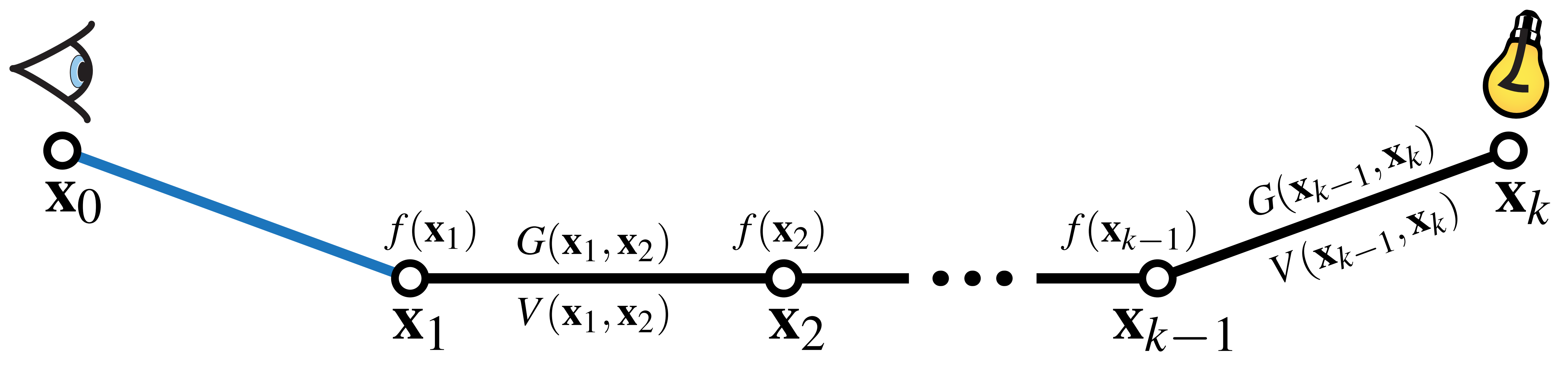
In order to solve the equation, we can construct paths between the camera and the light sources...

here you see all the terms defining the throughput of the path and as we mentioned previously, the main idea of instant radiosity is to collapse ... this ... subpath into a virtual light source.

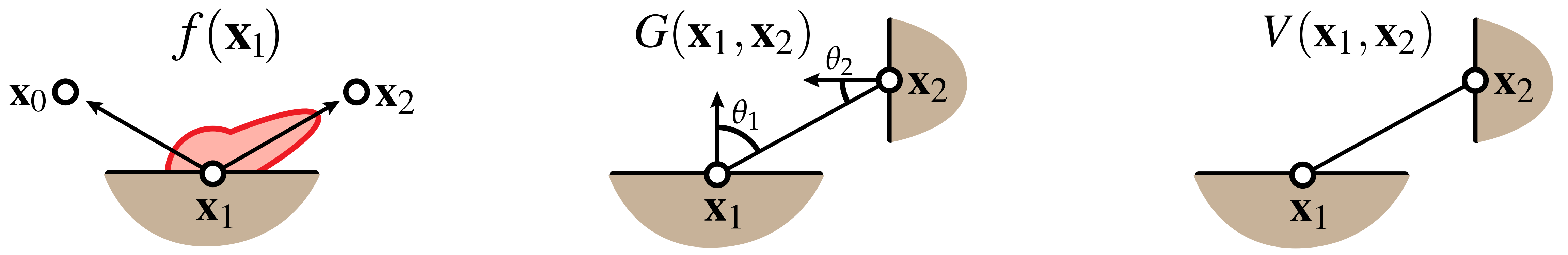
And once we have these virtual lights, we just need to connect the shaded points to them and evaluate the four terms defined with respect to the red connection.

Lighting with VPLs (assuming no media)

Paths between camera and light sources



BRDF **Geometry term** **Visibility term**



$$G(\mathbf{x}_1,\mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\|\mathbf{x}_1-\mathbf{x}_2\|^2}$$

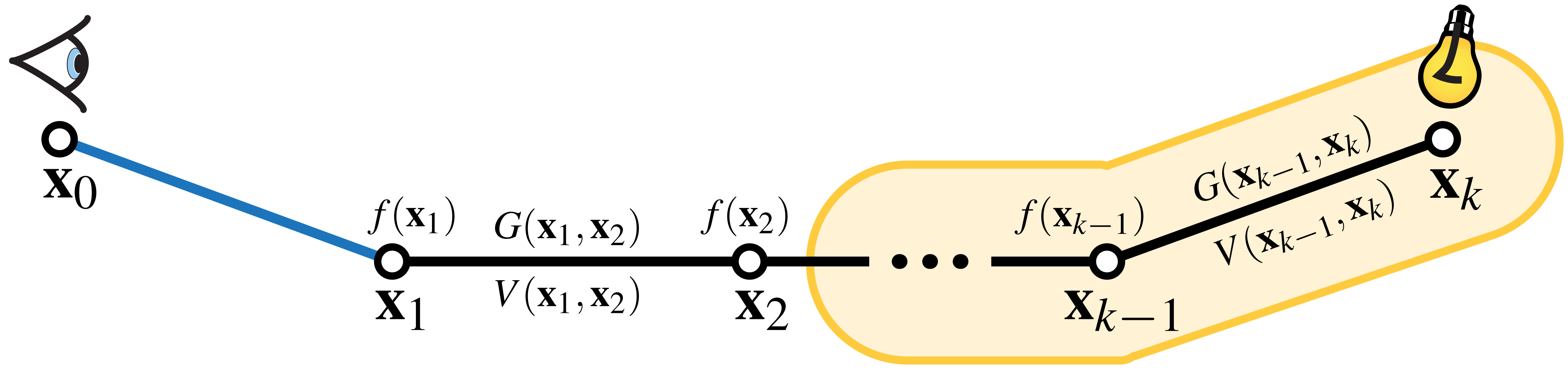
In order to solve the equation, we can construct paths between the camera and the light sources...

here you see all the terms defining the throughput of the path and as we mentioned previously, the main idea of instant radiosity is to collapse ... this ... subpath into a virtual light source.

And once we have these virtual lights, we just need to connect the shaded points to them and evaluate the four terms defined with respect to the red connection.

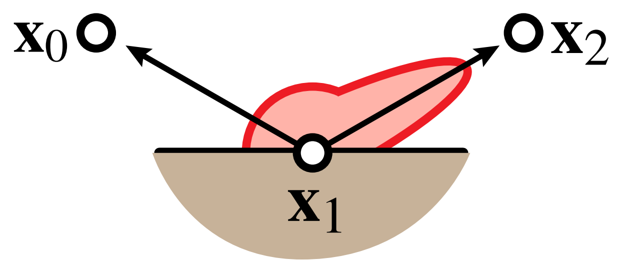
Lighting with VPLs (assuming no media)

Paths between camera and light sources



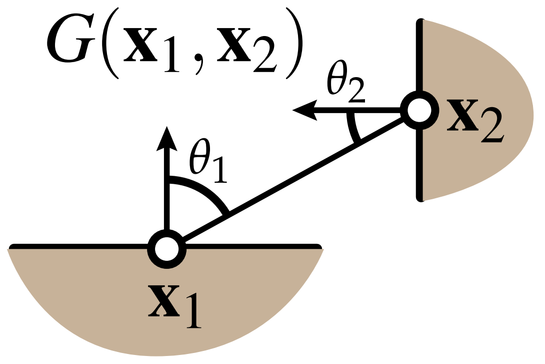
BRDF

$f(\mathbf{x}_1)$



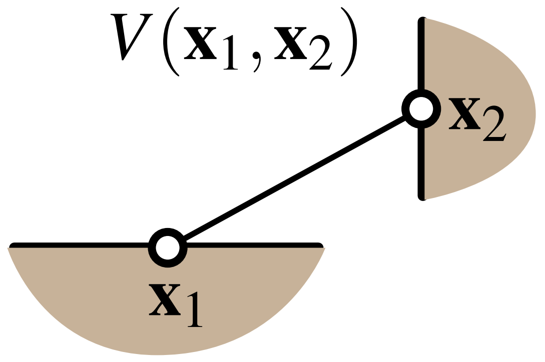
Geometry term

$G(\mathbf{x}_1,\mathbf{x}_2)$



Visibility term

$V(\mathbf{x}_1,\mathbf{x}_2)$



$$G(\mathbf{x}_1,\mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

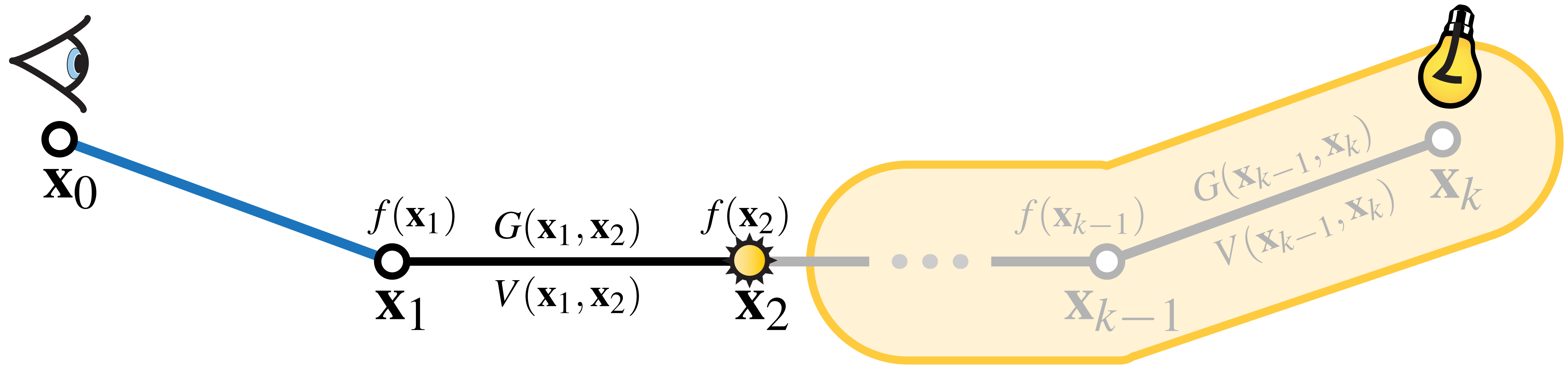
In order to solve the equation, we can construct paths between the camera and the light sources...

here you see all the terms defining the throughput of the path and as we mentioned previously, the main idea of instant radiosity is to collapse ... this ... subpath into a virtual light source.

And once we have these virtual lights, we just need to connect the shaded points to them and evaluate the four terms defined with respect to the red connection.

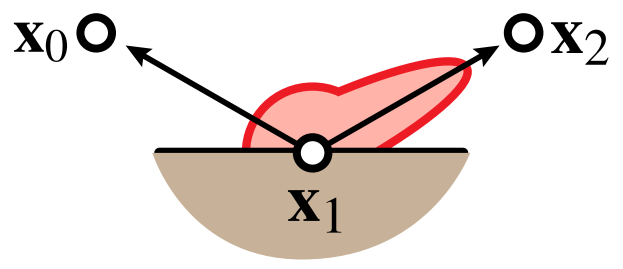
Lighting with VPLs (assuming no media)

Paths between camera and light sources



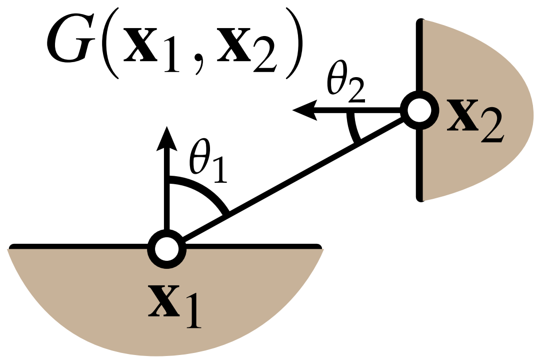
BRDF

$f(\mathbf{x}_1)$



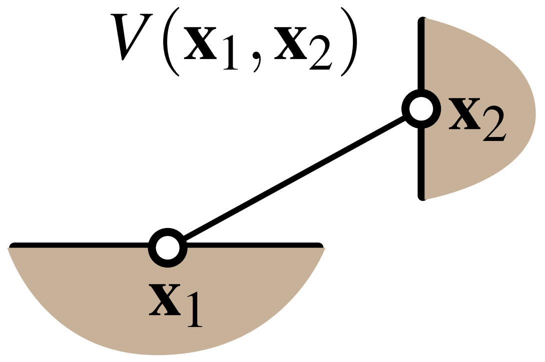
Geometry term

$G(\mathbf{x}_1,\mathbf{x}_2)$



Visibility term

$V(\mathbf{x}_1,\mathbf{x}_2)$



$$G(\mathbf{x}_1,\mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

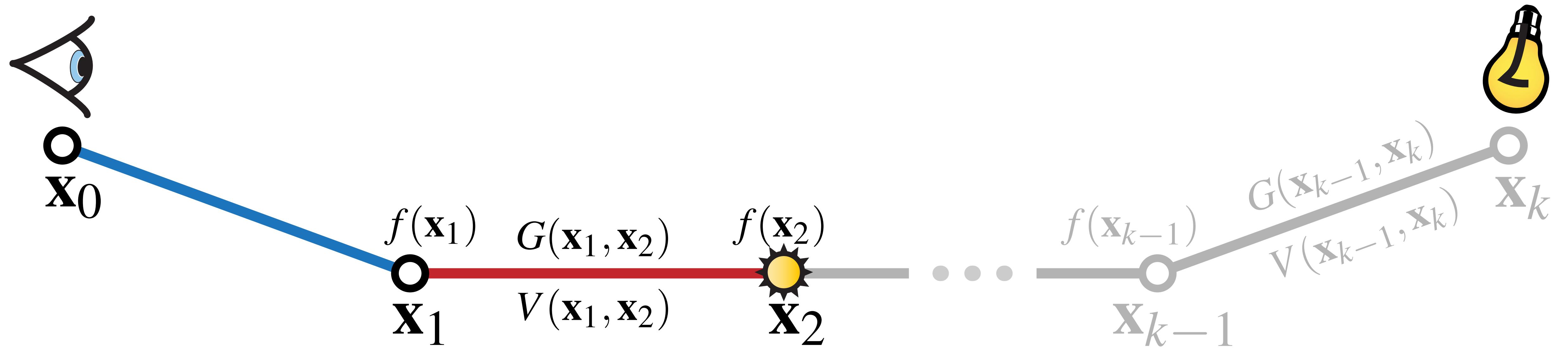
In order to solve the equation, we can construct paths between the camera and the light sources...

here you see all the terms defining the throughput of the path and as we mentioned previously, the main idea of instant radiosity is to collapse ... this ... subpath into a virtual light source.

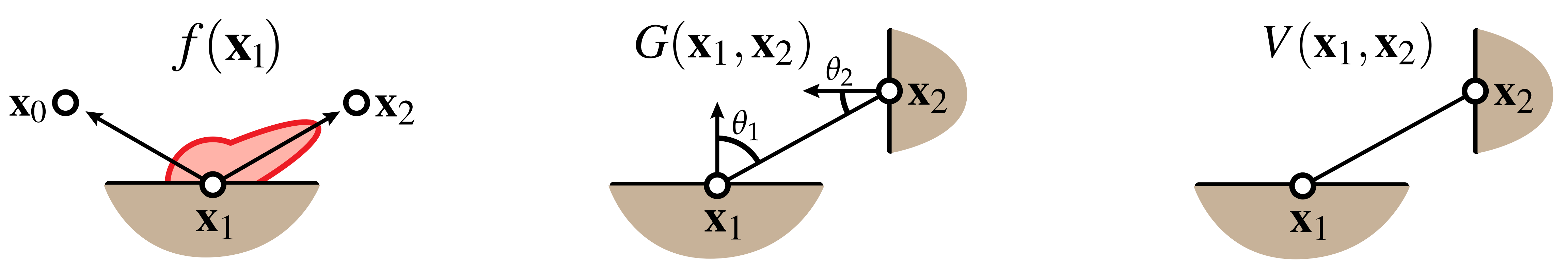
And once we have these virtual lights, we just need to connect the shaded points to them and evaluate the four terms defined with respect to the red connection.

Lighting with VPLs (assuming no media)

Paths between camera and light sources



BRDF | **Geometry term** | **Visibility term**



$$G(\mathbf{x}_1,\mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\|\mathbf{x}_1-\mathbf{x}_2\|^2}$$

In order to solve the equation, we can construct paths between the camera and the light sources...

here you see all the terms defining the throughput of the path and as we mentioned previously, the main idea of instant radiosity is to collapse ... this ... subpath into a virtual light source.

And once we have these virtual lights, we just need to connect the shaded points to them and evaluate the four terms defined with respect to the red connection.

Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2)\, V(\mathbf{x}_1, \mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

So going back to the rendering equation we can rewrite it for the special case of lighting with VPLs:

the integral is replaced by a sum over the VPLs; the BRDF, the geometry and the visibility terms remain the same,
except that x_2 is the point of the VPL, and the outgoing radiance from point x_2 towards point x_1 is replaced by the BRDF at the VPL and its flux, which effectively hides the recursion.

Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2)\, V(\mathbf{x}_1, \mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

$$L(\mathbf{x}_1 \to \mathbf{x}_0) \approx L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \sum_{i=1}^{N} f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2^i)\, V(\mathbf{x}_1, \mathbf{x}_2^i)\, f(\mathbf{x}_2^i)\, \Phi_i$$

Approximation using VPLs

- $\mathbf{x}_2^i$ position of i^{th} VPL
- Φ_i “flux” of i^{th} VPL

So going back to the rendering equation we can rewrite it for the special case of lighting with VPLs:

the integral is replaced by a sum over the VPLs; the BRDF, the geometry and the visibility terms remain the same,
except that x_2 is the point of the VPL, and the outgoing radiance from point x_2 towards point x_1 is replaced by the BRDF at the VPL and its flux, which effectively hides the recursion.

Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1\to\mathbf{x}_0) \;=\; L_e(\mathbf{x}_1\to\mathbf{x}_0) + \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1,\mathbf{x}_2)\, V(\mathbf{x}_1,\mathbf{x}_2)\, L(\mathbf{x}_2\to\mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

$$L(\mathbf{x}_1\to\mathbf{x}_0) \;\approx\; L_e(\mathbf{x}_1\to\mathbf{x}_0) + \sum_{i=1}^{N} f(\mathbf{x}_1)\, G(\mathbf{x}_1,\mathbf{x}_2^i)\, V(\mathbf{x}_1,\mathbf{x}_2^i)\, f(\mathbf{x}_2^i)\, \Phi_i$$

Approximation using VPLs

- $\mathbf{x}_2^i$ position of i^{th} VPL
- Φ_i "flux" of i^{th} VPL

So going back to the rendering equation we can rewrite it for the special case of lighting with VPLs:

the integral is replaced by a sum over the VPLs; the BRDF, the geometry and the visibility terms remain the same,
except that x_2 is the point of the VPL, and the outgoing radiance from point x_2 towards point x_1 is replaced by the BRDF at the VPL and its flux, which effectively hides the recursion.

Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \int_A \boxed{f(\mathbf{x}_1)}\, G(\mathbf{x}_1, \mathbf{x}_2)\, V(\mathbf{x}_1, \mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

$$L(\mathbf{x}_1 \to \mathbf{x}_0) \approx L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \sum_{i=1}^{N} \boxed{f(\mathbf{x}_1)}\, G(\mathbf{x}_1, \mathbf{x}_2^i)\, V(\mathbf{x}_1, \mathbf{x}_2^i)\, f(\mathbf{x}_2^i)\, \Phi_i$$

Approximation using VPLs

- $\mathbf{x}_2^i$ position of i^{th} VPL
- Φ_i "flux" of i^{th} VPL

So going back to the rendering equation we can rewrite it for the special case of lighting with VPLs:

the integral is replaced by a sum over the VPLs; the BRDF, the geometry and the visibility terms remain the same,
except that x_2 is the point of the VPL, and the outgoing radiance from point x_2 towards point x_1 is replaced by the BRDF at the VPL and its flux, which effectively hides the recursion.

Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \int_A f(\mathbf{x}_1)\, \boxed{G(\mathbf{x}_1, \mathbf{x}_2)}\, V(\mathbf{x}_1, \mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

$$L(\mathbf{x}_1 \to \mathbf{x}_0) \approx L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \sum_{i=1}^{N} f(\mathbf{x}_1)\, \boxed{G(\mathbf{x}_1, \mathbf{x}_2^i)}\, V(\mathbf{x}_1, \mathbf{x}_2^i)\, f(\mathbf{x}_2^i)\, \Phi_i$$

Approximation using VPLs

- $\mathbf{x}_2^i$ position of i^{th} VPL
- Φ_i "flux" of i^{th} VPL

So going back to the rendering equation we can rewrite it for the special case of lighting with VPLs:

the integral is replaced by a sum over the VPLs; the BRDF, the geometry and the visibility terms remain the same,
except that x_2 is the point of the VPL, and the outgoing radiance from point x_2 towards point x_1 is replaced by the BRDF at the VPL and its flux, which effectively hides the recursion.

Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2)\, \boxed{V(\mathbf{x}_1, \mathbf{x}_2)}\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

$$L(\mathbf{x}_1 \to \mathbf{x}_0) \approx L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \sum_{i=1}^{N} f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2^i)\, \boxed{V(\mathbf{x}_1, \mathbf{x}_2^i)}\, f(\mathbf{x}_2^i)\, \Phi_i$$

Approximation using VPLs

- $\mathbf{x}_2^i$ position of i^{th} VPL
- Φ_i "flux" of i^{th} VPL

So going back to the rendering equation we can rewrite it for the special case of lighting with VPLs:

the integral is replaced by a sum over the VPLs; the BRDF, the geometry and the visibility terms remain the same,
except that x_2 is the point of the VPL, and the outgoing radiance from point x_2 towards point x_1 is replaced by the BRDF at the VPL and its flux, which effectively hides the recursion.

Lighting with VPLs (assuming no media)

Rendering Equation (area formulation)

$$L(\mathbf{x}_1 \to \mathbf{x}_0) \; = \; L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2)\, V(\mathbf{x}_1, \mathbf{x}_2)\, \boxed{L(\mathbf{x}_2 \to \mathbf{x}_1)}\, \mathrm{d}A(\mathbf{x}_2)$$

$$L(\mathbf{x}_1 \to \mathbf{x}_0) \; \approx \; L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \sum_{i=1}^{N} f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2^i)\, V(\mathbf{x}_1, \mathbf{x}_2^i)\, \boxed{f(\mathbf{x}_2^i)\, \Phi_i}$$

recursion is hidden
in the generation of VPLs

Approximation using VPLs

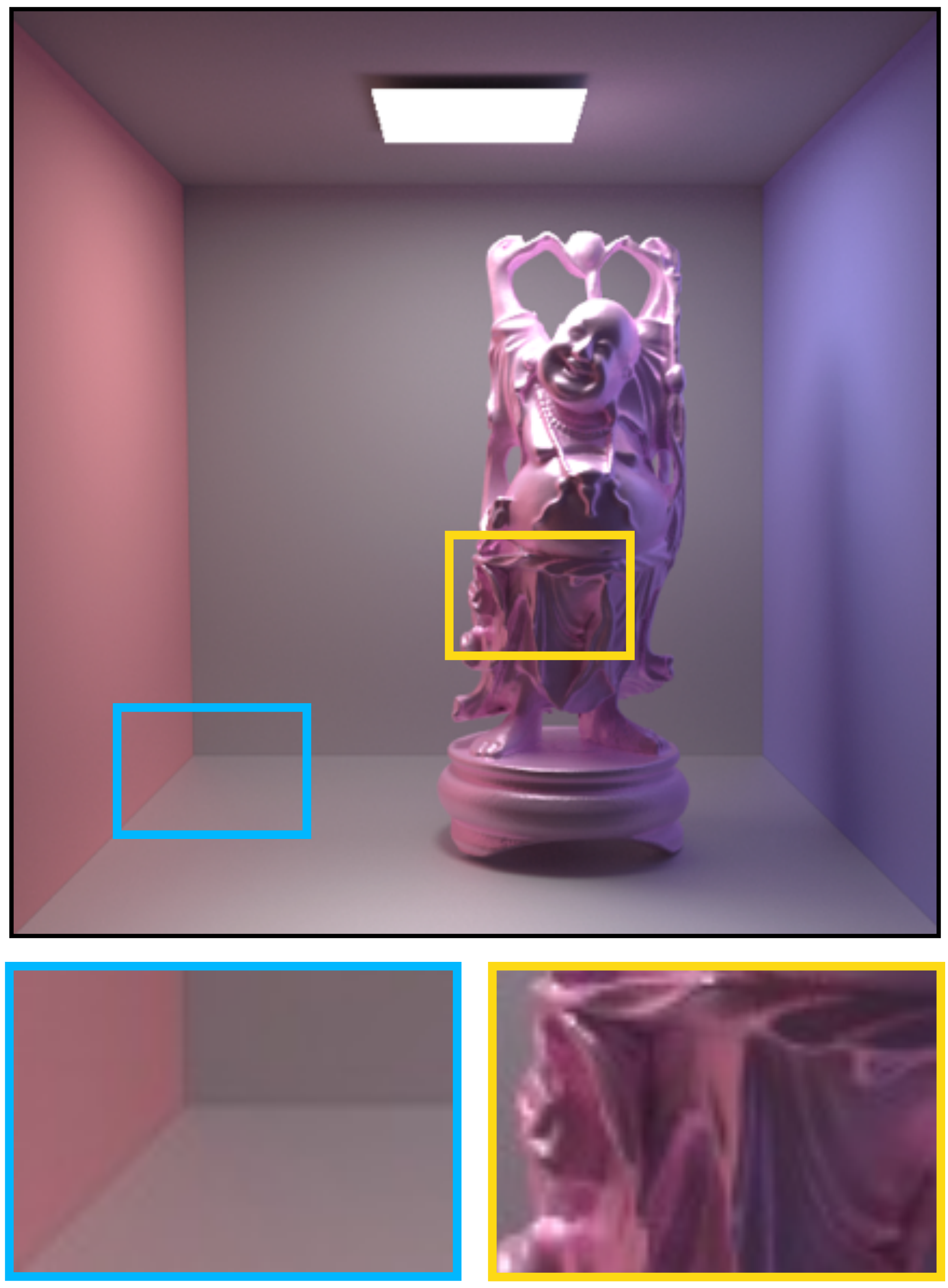
- $\mathbf{x}_2^i$ position of i^{th} VPL
- Φ_i “flux” of i^{th} VPL

So going back to the rendering equation we can rewrite it for the special case of lighting with VPLs:

the integral is replaced by a sum over the VPLs; the BRDF, the geometry and the visibility terms remain the same,
except that x_2 is the point of the VPL, and the outgoing radiance from point x_2 towards point x_1 is replaced by the BRDF at the VPL and its flux, which effectively hides the recursion.

Lighting with VPLs (assuming no media)

Reference



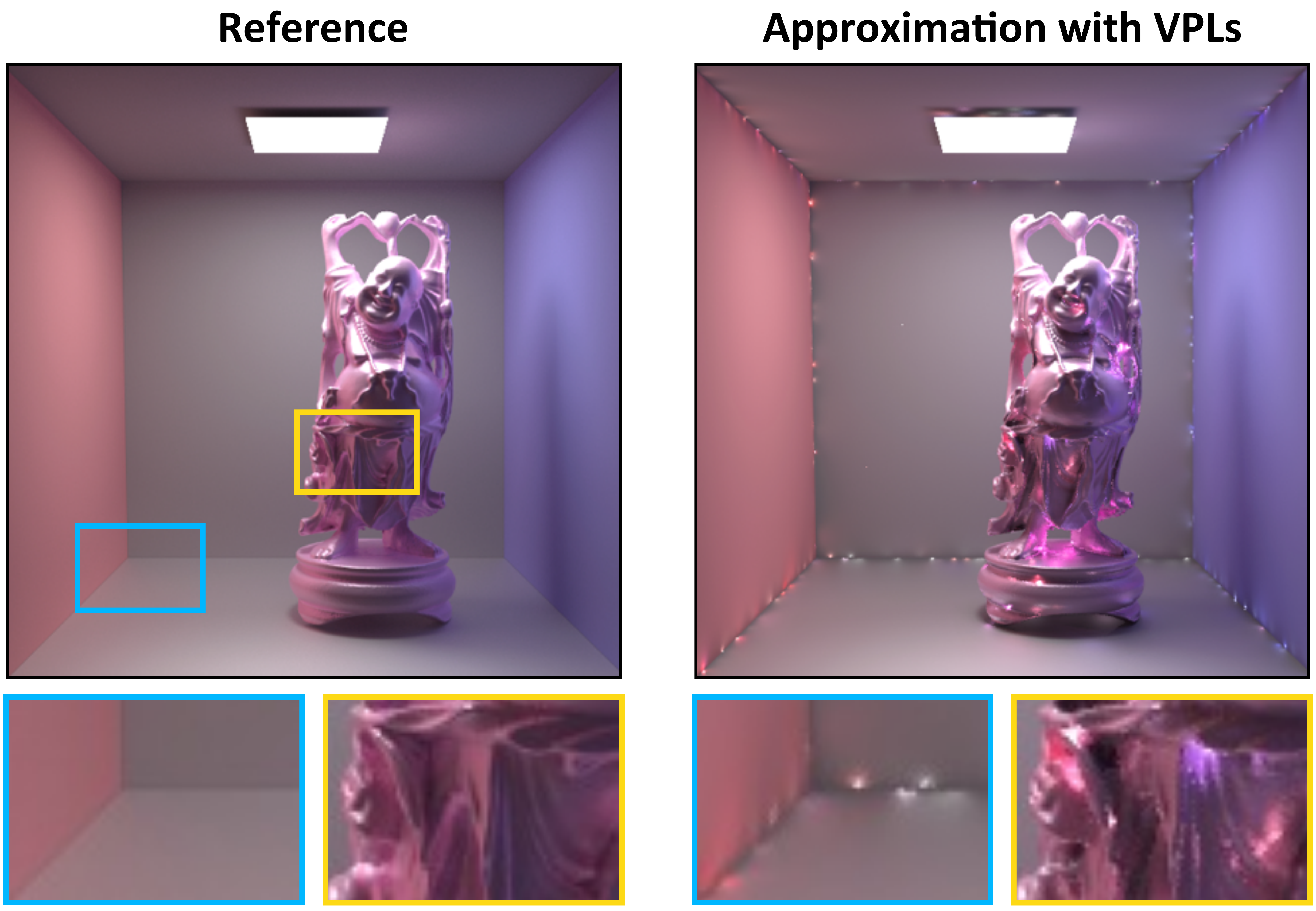
So let's have a look at an example rendering. Here we have a path traced image for reference and an image rendered using VPLs.

And you can clearly see some splotchy artifacts. There are two reasons why we have these splotches.

The first and the more crucial one is the geometry term, which has a squared distance between the two points in the denominator. As the distance between the shading point and the VPL becomes smaller, the value of the geometry term will grow higher and this is emphasized by the fact that we even square the distance.

So this is the reason why the estimator can have very large values in some cases, the other reason why we see the splotches very clearly is that in this case we connect all the shading points to the SAME set of VPLs. We get noise-free results, which is one of the biggest advantages of instant radiosity, but the estimation error is still there in the form of structured artifacts.

Lighting with VPLs (assuming no media)



So let's have a look at an example rendering. Here we have a path traced image for reference and an image rendered using VPLs.

And you can clearly see some splotchy artifacts. There are two reasons why we have these splotches.

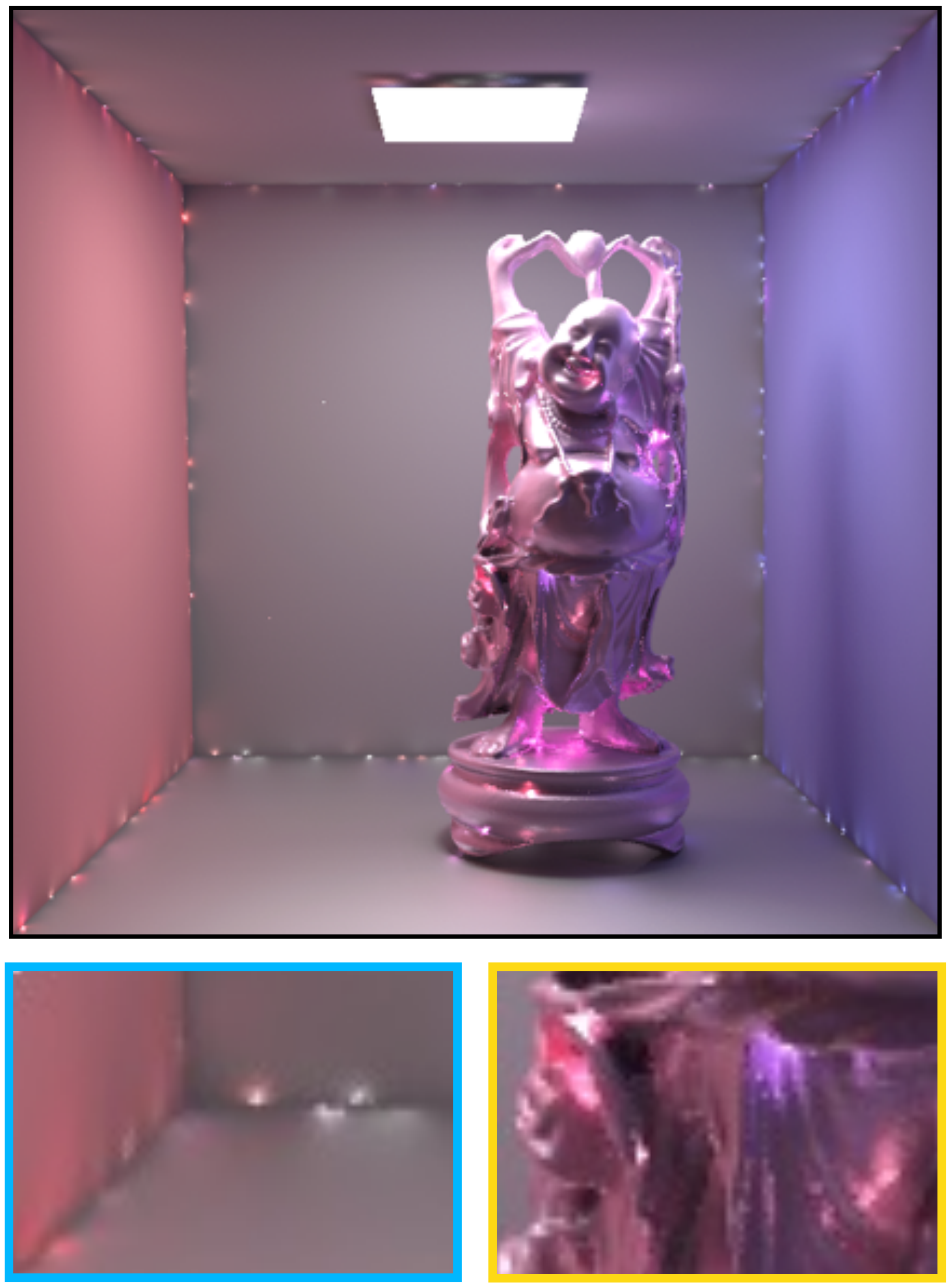
The first and the more crucial one is the geometry term, which has a squared distance between the two points in the denominator. As the distance between the shading point and the VPL becomes smaller, the value of the geometry term will grow higher and this is emphasized by the fact that we even square the distance.

So this is the reason why the estimator can have very large values in some cases, the other reason why we see the splotches very clearly is that in this case we connect all the shading points to the SAME set of VPLs. We get noise-free results, which is one of the biggest advantages of instant radiosity, but the estimation error is still there in the form of structured artifacts.

Lighting with VPLs (assuming no media)

Splotches!!!

Approximation with VPLs



So let’s have a look at an example rendering. Here we have a path traced image for reference and an image rendered using VPLs.

And you can clearly see some splotchy artifacts. There are two reasons why we have these splotches.

The first and the more crucial one is the geometry term, which has a squared distance between the two points in the denominator. As the distance between the shading point and the VPL becomes smaller, the value of the geometry term will grow higher and this is emphasized by the fact that we even square the distance.

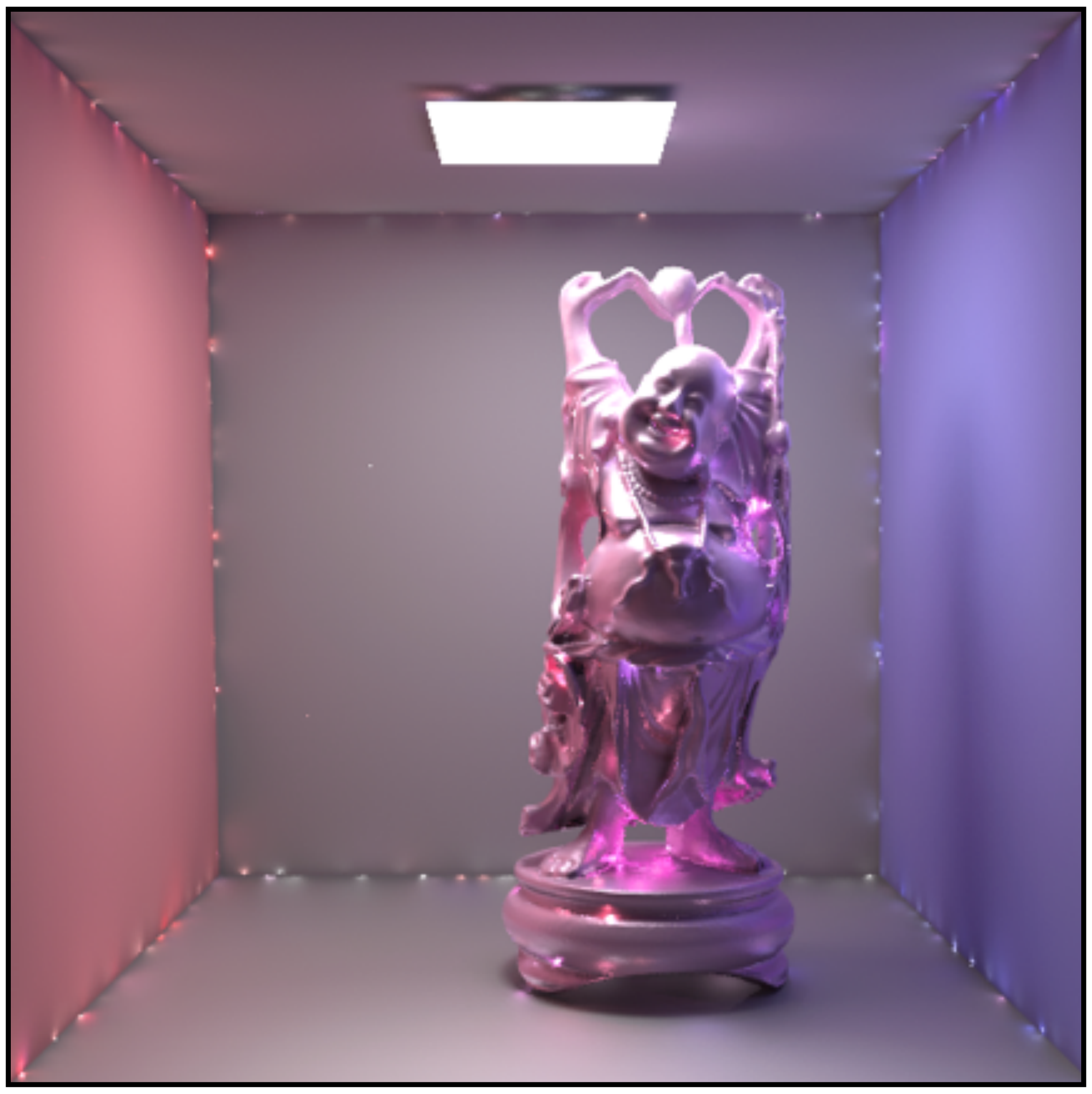
So this is the reason why the estimator can have very large values in some cases, the other reason why we see the splotches very clearly is that in this case we connect all the shading points to the SAME set of VPLs. We get noise-free results, which is one of the biggest advantages of instant radiosity, but the estimation error is still there in the form of structured artifacts.

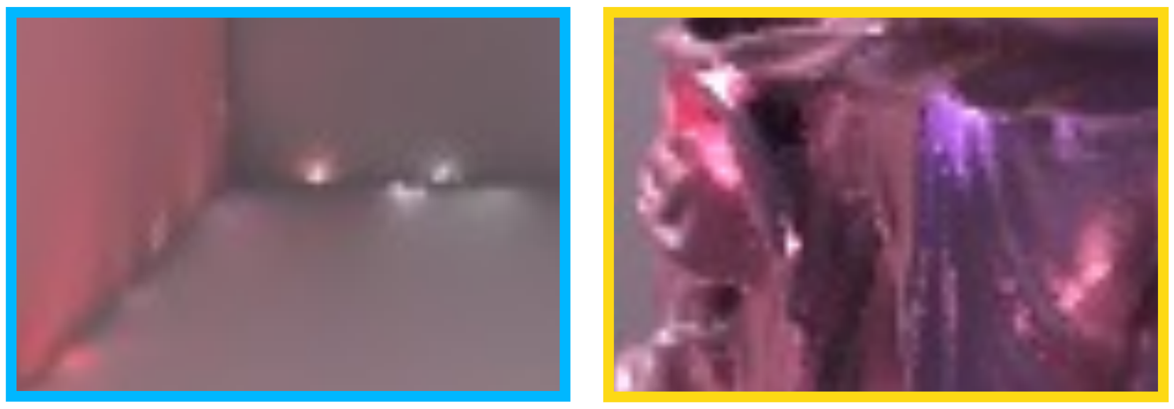
Lighting with VPLs (assuming no media)

Splotches!!!

Reasons:

Approximation with VPLs





So let's have a look at an example rendering. Here we have a path traced image for reference and an image rendered using VPLs.

And you can clearly see some splotchy artifacts. There are two reasons why we have these splotches.

The first and the more crucial one is the geometry term, which has a squared distance between the two points in the denominator. As the distance between the shading point and the VPL becomes smaller, the value of the geometry term will grow higher and this is emphasized by the fact that we even square the distance.

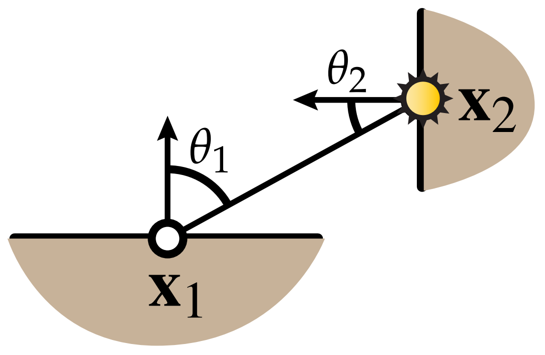
So this is the reason why the estimator can have very large values in some cases, the other reason why we see the splotches very clearly is that in this case we connect all the shading points to the SAME set of VPLs. We get noise-free results, which is one of the biggest advantages of instant radiosity, but the estimation error is still there in the form of structured artifacts.

Lighting with VPLs (assuming no media)

Splotches!!!

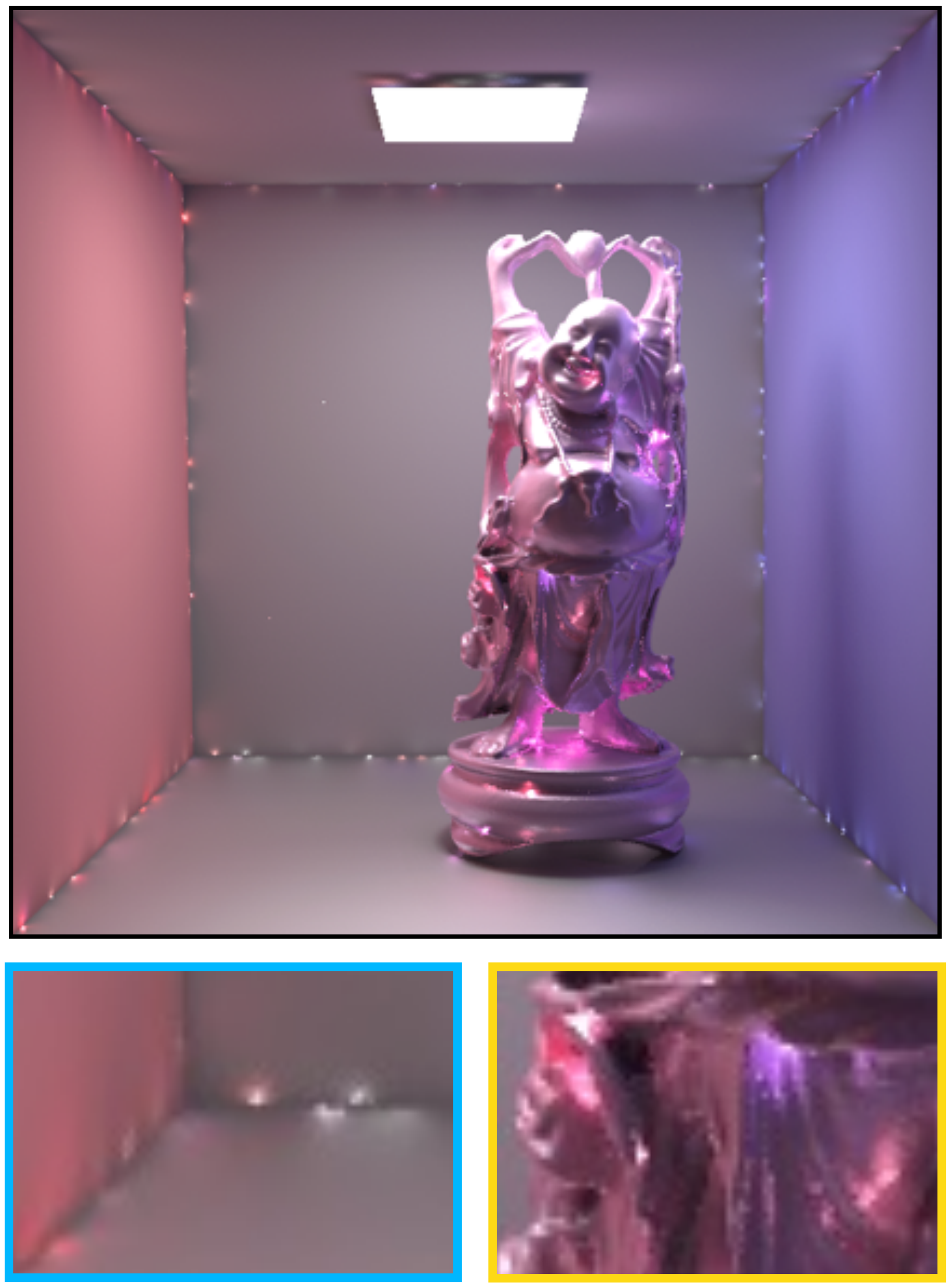
Reasons:

- geometry term



$$G(\mathbf{x}_1, \mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

Approximation with VPLs



So let's have a look at an example rendering. Here we have a path traced image for reference and an image rendered using VPLs.

And you can clearly see some splotchy artifacts. There are two reasons why we have these splotches.

The first and the more crucial one is the geometry term, which has a squared distance between the two points in the denominator. As the distance between the shading point and the VPL becomes smaller, the value of the geometry term will grow higher and this is emphasized by the fact that we even square the distance.

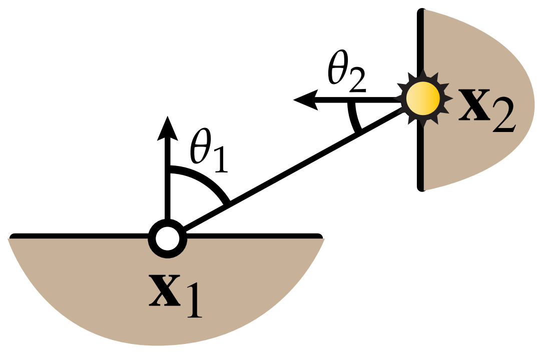
So this is the reason why the estimator can have very large values in some cases, the other reason why we see the splotches very clearly is that in this case we connect all the shading points to the SAME set of VPLs. We get noise-free results, which is one of the biggest advantages of instant radiosity, but the estimation error is still there in the form of structured artifacts.

Lighting with VPLs (assuming no media)

Splotches!!!

Reasons:

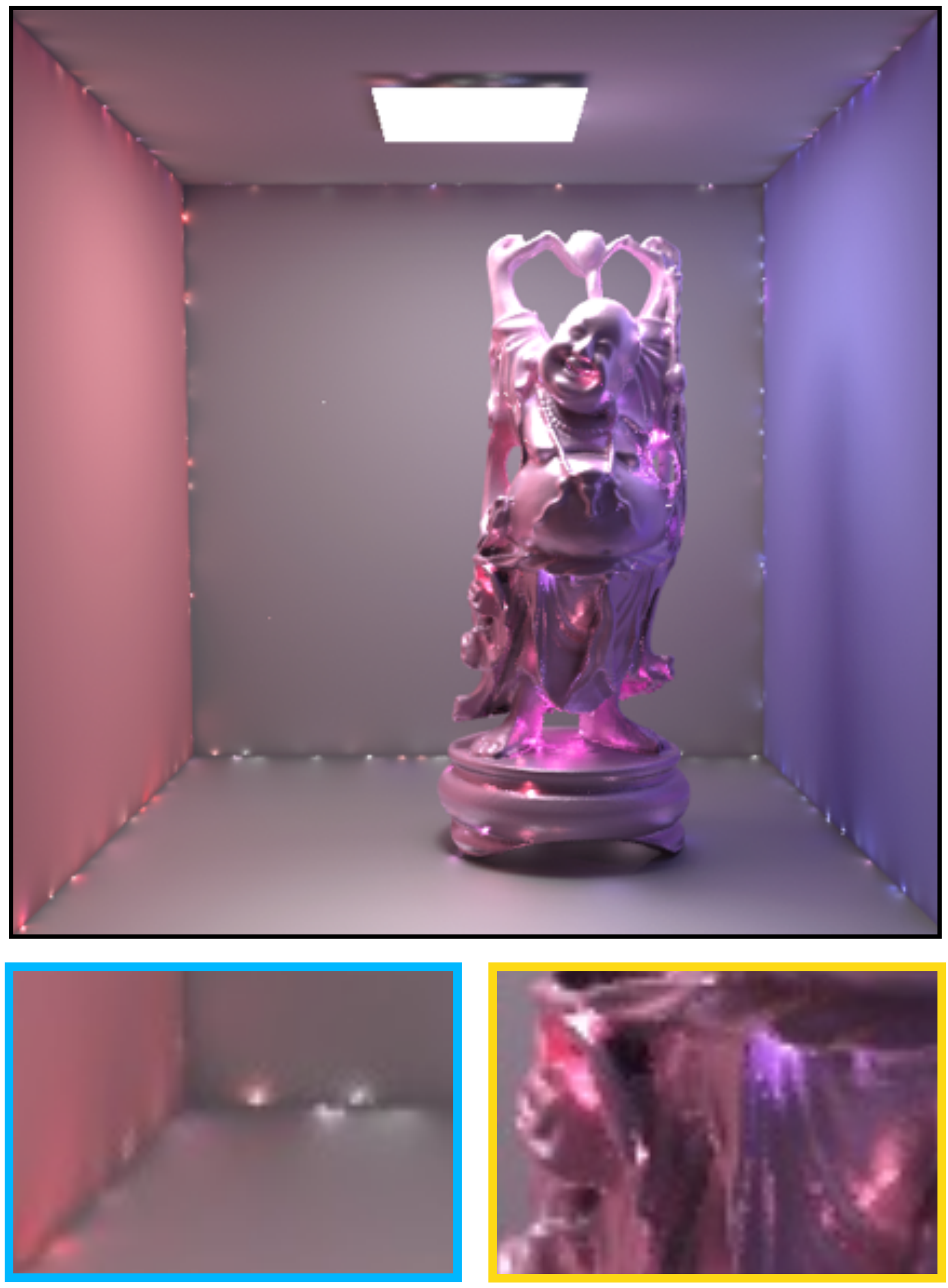
- geometry term



$$G(\mathbf{x}_1,\mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\boxed{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}}$$

singularity

Approximation with VPLs



So let's have a look at an example rendering. Here we have a path traced image for reference and an image rendered using VPLs.

And you can clearly see some splotchy artifacts. There are two reasons why we have these splotches.

The first and the more crucial one is the geometry term, which has a squared distance between the two points in the denominator. As the distance between the shading point and the VPL becomes smaller, the value of the geometry term will grow higher and this is emphasized by the fact that we even square the distance.

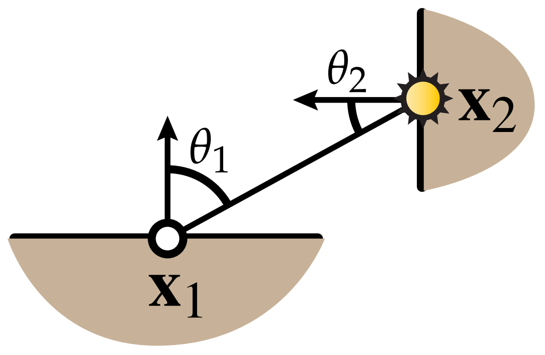
So this is the reason why the estimator can have very large values in some cases, the other reason why we see the splotches very clearly is that in this case we connect all the shading points to the SAME set of VPLs. We get noise-free results, which is one of the biggest advantages of instant radiosity, but the estimation error is still there in the form of structured artifacts.

Lighting with VPLs (assuming no media)

Splotches!!!

Reasons:

- geometry term

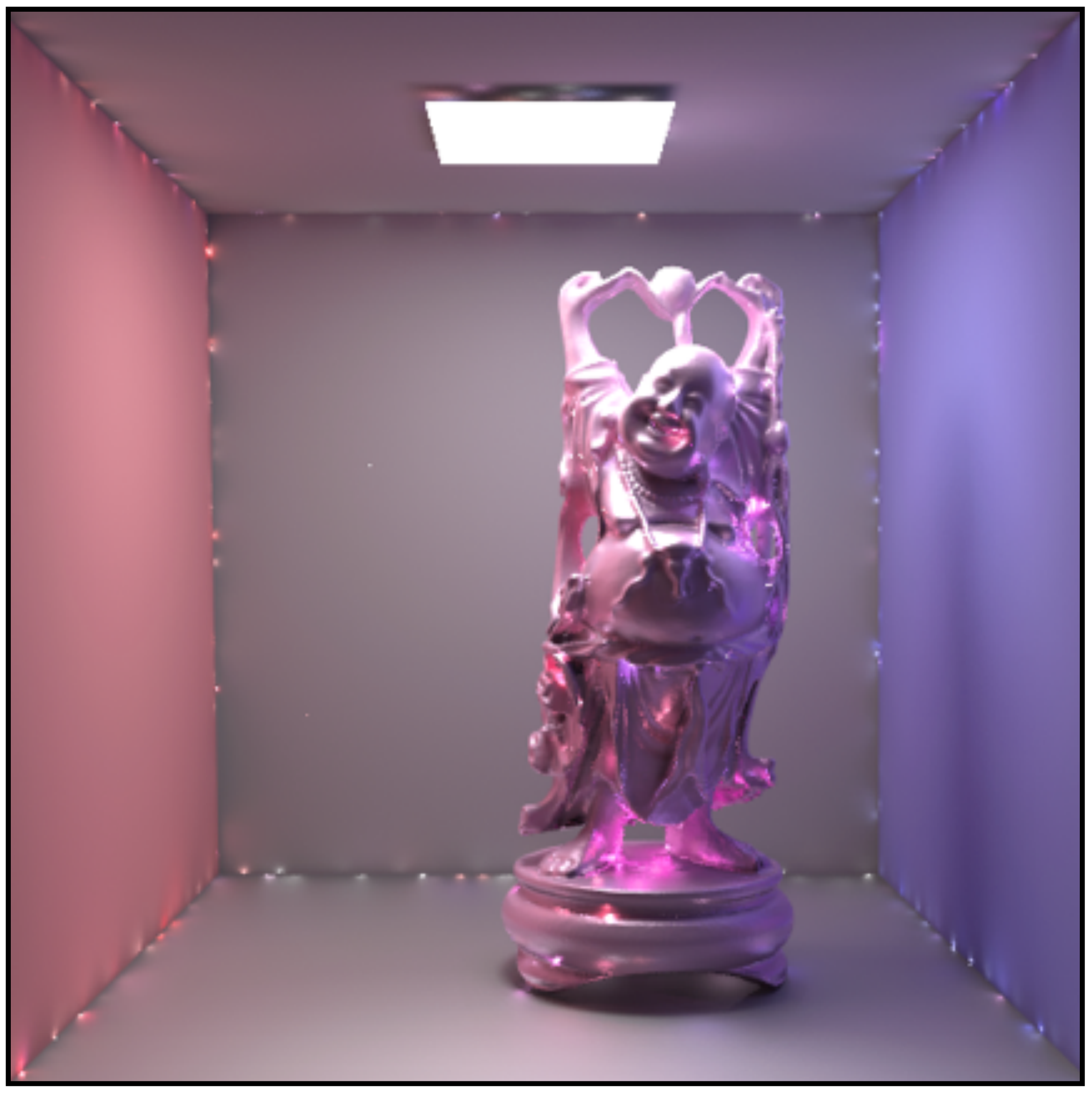


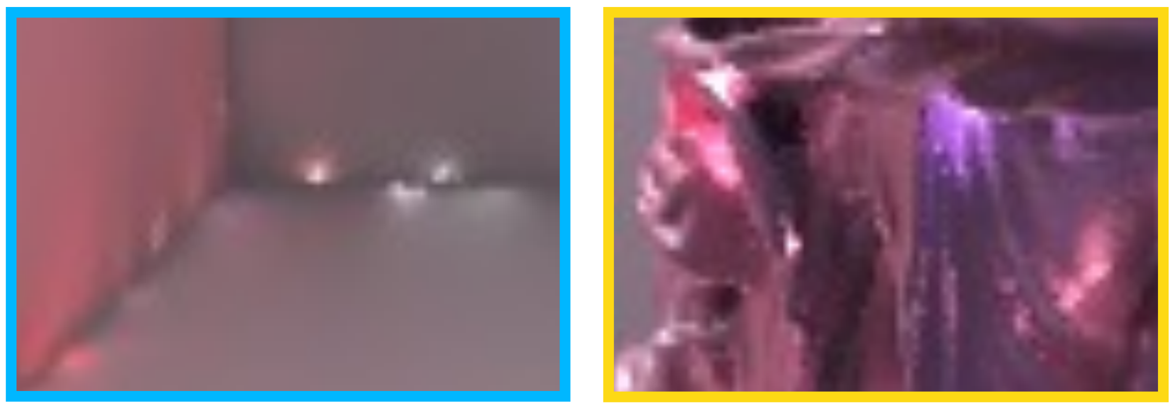
$$G(\mathbf{x}_1,\mathbf{x}_2) = \frac{\cos(\theta_1)\cos(\theta_2)}{\boxed{\|\mathbf{x}_1-\mathbf{x}_2\|^2}}$$

singularity

- correlation in the estimator
  - all points are lit by the same set of VPLs

Approximation with VPLs





So let's have a look at an example rendering. Here we have a path traced image for reference and an image rendered using VPLs.

And you can clearly see some splotchy artifacts. There are two reasons why we have these splotches.

The first and the more crucial one is the geometry term, which has a squared distance between the two points in the denominator. As the distance between the shading point and the VPL becomes smaller, the value of the geometry term will grow higher and this is emphasized by the fact that we even square the distance.

So this is the reason why the estimator can have very large values in some cases, the other reason why we see the splotches very clearly is that in this case we connect all the shading points to the SAME set of VPLs. We get noise-free results, which is one of the biggest advantages of instant radiosity, but the estimation error is still there in the form of structured artifacts.

Lighting with VPLs

Splotches!!!

Now the question is how to get rid off these artifacts.

There have been two major approaches developed for this over the past few years.

The first one is to bound the geometry term... in other words, we clamp its value to a certain threshold.
This removes the splotches, but at the same time we loose some energy and the rendered images will become darker. There are several publications trying to recover this missing energy.

The other approach is to distribute the energy over a certain area or volume, so that it is no longer concentrated at single points. So instead of removing the energy we rather spread it spatially.

So let's first have a look at techniques that start by bounding the geometry term.

Lighting with VPLs

Splotches!!!

Solutions:

Now the question is how to get rid off these artifacts.

There have been two major approaches developed for this over the past few years.

The first one is to bound the geometry term… in other words, we clamp its value to a certain threshold.
This removes the splotches, but at the same time we loose some energy and the rendered images will become darker. There are several publications trying to recover this missing energy.

The other approach is to distribute the energy over a certain area or volume, so that it is no longer concentrated at single points. So instead of removing the energy we rather spread it spatially.

So let’s first have a look at techniques that start by bounding the geometry term.

Lighting with VPLs

Splotches!!!

Solutions:

1. **Bound the geometry term**
   - removes energy, darkens the image
   - to get unbiased results, we need to compensate for the bounding

     [Kollig and Keller 2004], [Raab et al. 2008], [Davidovič et al. 2010], [Novák et al. 2011], [Engelhardt et al. 2012]

Now the question is how to get rid off these artifacts.

There have been two major approaches developed for this over the past few years.

The first one is to bound the geometry term... in other words, we clamp its value to a certain threshold.
This removes the splotches, but at the same time we loose some energy and the rendered images will become darker. There are several publications trying to recover this missing energy.

The other approach is to distribute the energy over a certain area or volume, so that it is no longer concentrated at single points. So instead of removing the energy we rather spread it spatially.

So let's first have a look at techniques that start by bounding the geometry term.

Lighting with VPLs

Splotches!!!

Solutions:

1. **Bound the geometry term**
   - removes energy, darkens the image
   - to get unbiased results, we need to compensate for the bounding

   [Kollig and Keller 2004], [Raab et al. 2008], [Davidovič et al. 2010], [Novák et al. 2011], [Engelhardt et al. 2012]

2. **Distribute the flux of a VPL over area (volume)**
   - redistributes energy, blurs the illumination
   - to get consistent results, progressively reduce the blurring

   [Hašan et al. 2009], [Novák et al. 2012a], [Novák et al. 2012b]

Now the question is how to get rid off these artifacts.

There have been two major approaches developed for this over the past few years.

The first one is to bound the geometry term... in other words, we clamp its value to a certain threshold.
This removes the splotches, but at the same time we loose some energy and the rendered images will become darker. There are several publications trying to recover this missing energy.

The other approach is to distribute the energy over a certain area or volume, so that it is no longer concentrated at single points. So instead of removing the energy we rather spread it spatially.

So let’s first have a look at techniques that start by bounding the geometry term.

Lighting with VPLs

Splotches!!!

Solutions:

1. **Bound the geometry term**
   - removes energy, darkens the image
   - to get unbiased results, we need to compensate for the bounding

   [Kollig and Keller 2004], [Raab et al. 2008], [Davidovič et al. 2010], [Novák et al. 2011], [Engelhardt et al. 2012]

2. **Distribute the flux of a VPL over area (volume)**
   - redistributes energy, blurs the illumination
   - to get consistent results, progressively reduce the blurring

   [Hašan et al. 2009], [Novák et al. 2012a], [Novák et al. 2012b]

Now the question is how to get rid off these artifacts.

There have been two major approaches developed for this over the past few years.

The first one is to bound the geometry term... in other words, we clamp its value to a certain threshold.
This removes the splotches, but at the same time we loose some energy and the rendered images will become darker. There are several publications trying to recover this missing energy.

The other approach is to distribute the energy over a certain area or volume, so that it is no longer concentrated at single points. So instead of removing the energy we rather spread it spatially.

So let’s first have a look at techniques that start by bounding the geometry term.

Bounding & Compensation

The idea is to simply clamp the value of G to a user-defined maximum value.

Here we define the bounded geometry term Gb as a minimum of the original G-term and the user-defined threshold b.

So this is an extremely simple, fast and popular technique, but as you will see, it has some major drawbacks.

Bounding & Compensation

Bounding the geometry term

- prevent G from being very high
- b - user-defined maximum value (bound)

$$G_b(\mathbf{x}_1, \mathbf{x}_2) = \min(G(\mathbf{x}_1, \mathbf{x}_2), b)$$

The idea is to simply clamp the value of G to a user-defined maximum value.

Here we define the bounded geometry term Gb as a minimum of the original G-term and the user-defined threshold b.

So this is an extremely simple, fast and popular technique, but as you will see, it has some major drawbacks.

Bounding & Compensation

Bounding the geometry term

- prevent G from being very high
- b - user-defined maximum value (bound)

$$G_b(\mathbf{x}_1, \mathbf{x}_2) = \min(G(\mathbf{x}_1, \mathbf{x}_2), b)$$

Advantages:

- extremely simple and fast

The idea is to simply clamp the value of G to a user-defined maximum value.

Here we define the bounded geometry term Gb as a minimum of the original G-term and the user-defined threshold b.

So this is an extremely simple, fast and popular technique, but as you will see, it has some major drawbacks.

Bounding & Compensation

Bounding the geometry term

- prevent G from being very high
- b - user-defined maximum value (bound)

$$G_b(\mathbf{x}_1, \mathbf{x}_2) = \min(G(\mathbf{x}_1, \mathbf{x}_2), b)$$

Advantages:

- extremely simple and fast

Disadvantages:

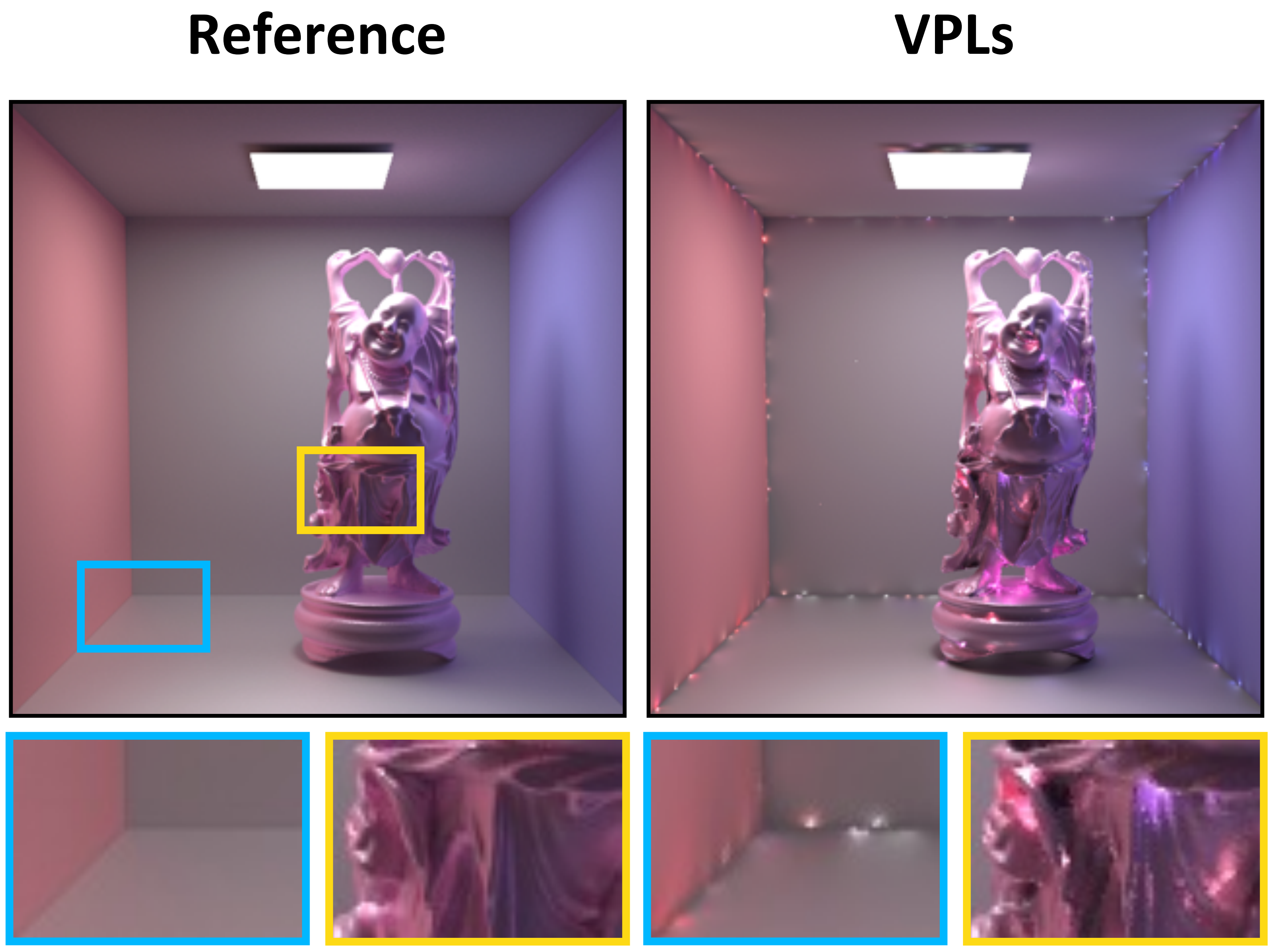
- removes energy, darkens the image

The idea is to simply clamp the value of G to a user-defined maximum value.

Here we define the bounded geometry term Gb as a minimum of the original G-term and the user-defined threshold b.

So this is an extremely simple, fast and popular technique, but as you will see, it has some major drawbacks.

Bounding & Compensation

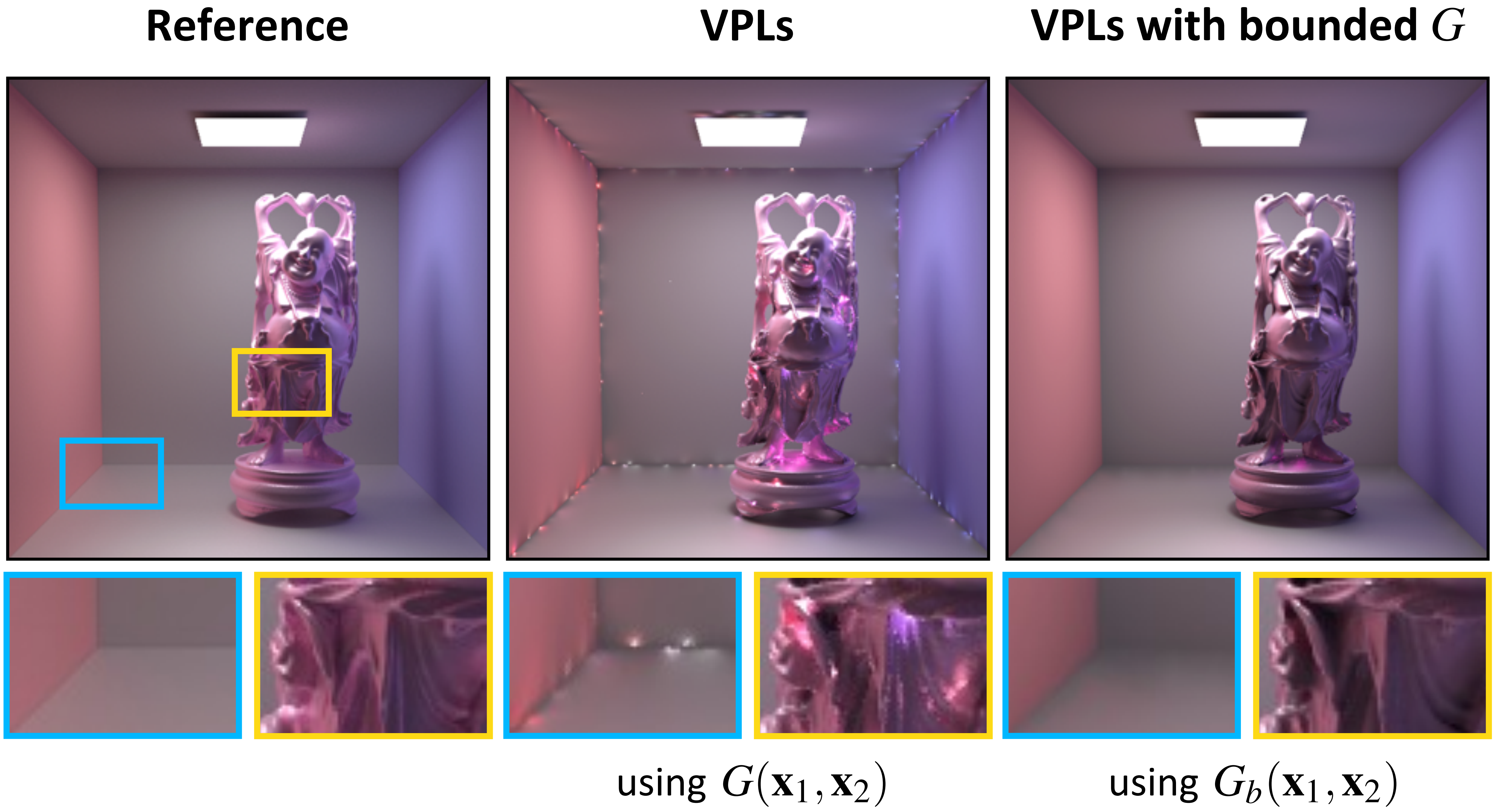


using $G(\mathbf{x}_1, \mathbf{x}_2)$

Here we have again the reference, the previous VPL rendering,... and on the right, a VPL rendering with the bounded geometry term.

You can see that it is darker around corners and cavities. And since this heavily degrades the quality of the rendered images, we shall now try to compensate for the missing energy and add it back.

Bounding & Compensation

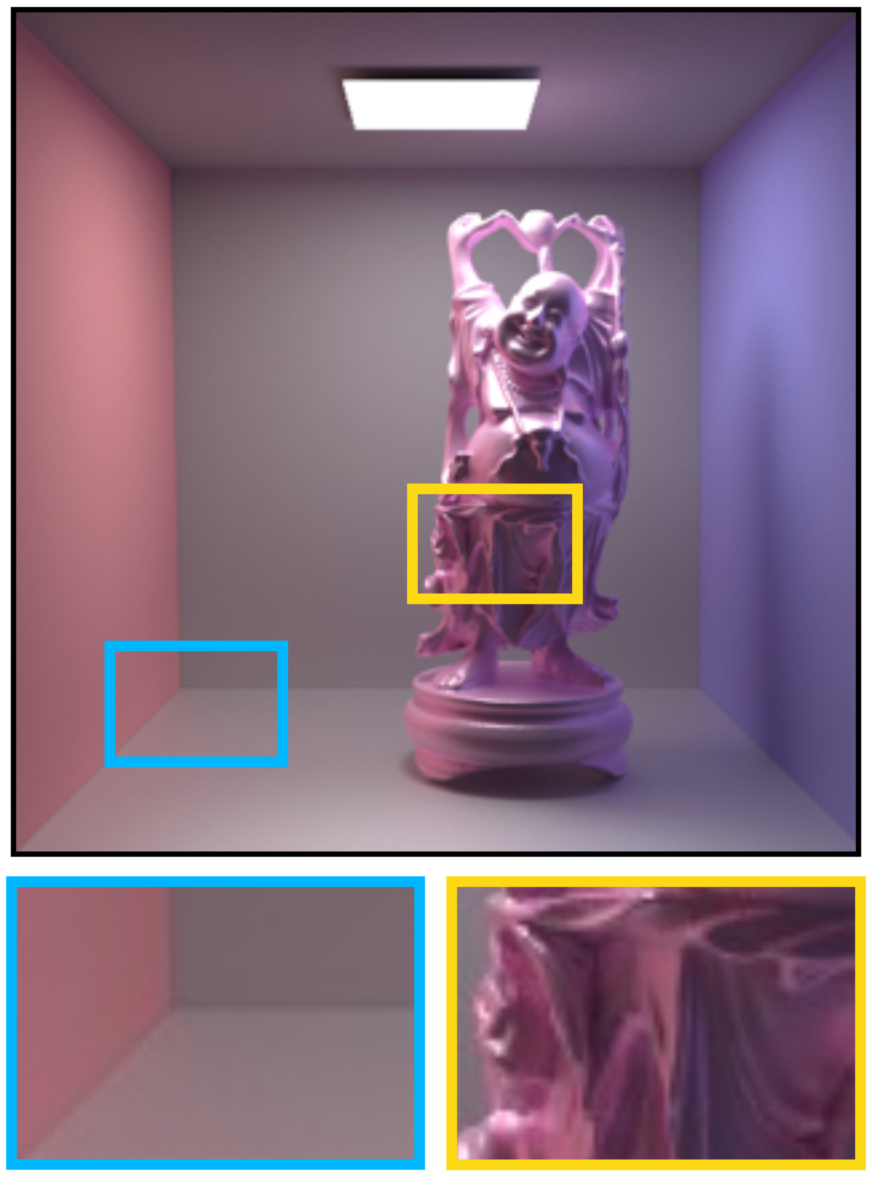

Reference
VPLs
VPLs with bounded G
using $G(\mathbf{x}_1, \mathbf{x}_2)$
using $G_b(\mathbf{x}_1, \mathbf{x}_2)$

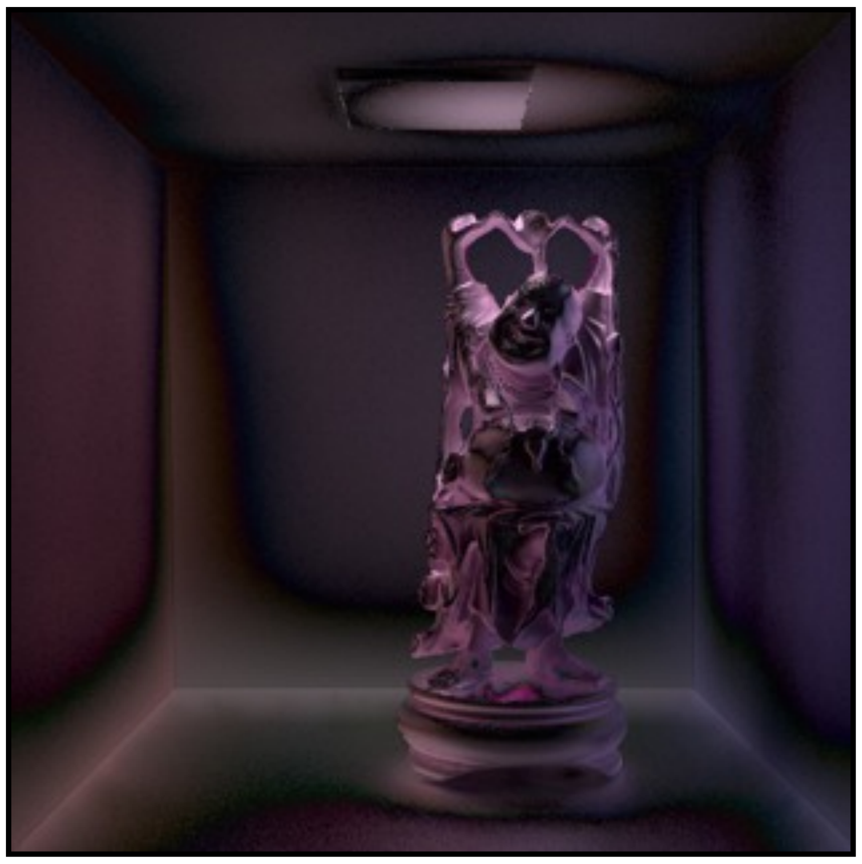
Here we have again the reference, the previous VPL rendering,... and on the right, a VPL rendering with the bounded geometry term.

You can see that it is darker around corners and cavities. And since this heavily degrades the quality of the rendered images, we shall now try to compensate for the missing energy and add it back.

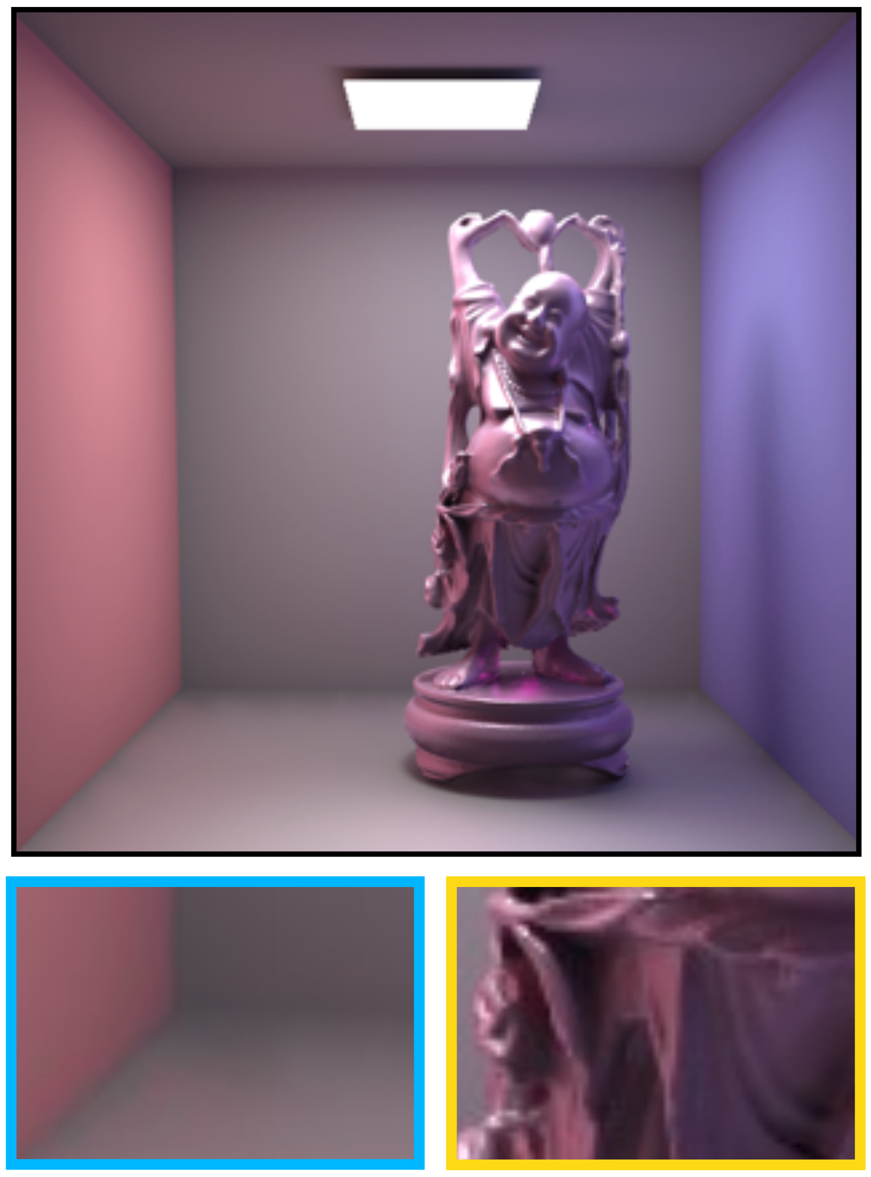
Bounding & Compensation

Reference ← Difference → VPLs with bounded G





“$G(\mathbf{x}_1,\mathbf{x}_2) - G_b(\mathbf{x}_1,\mathbf{x}_2)$”



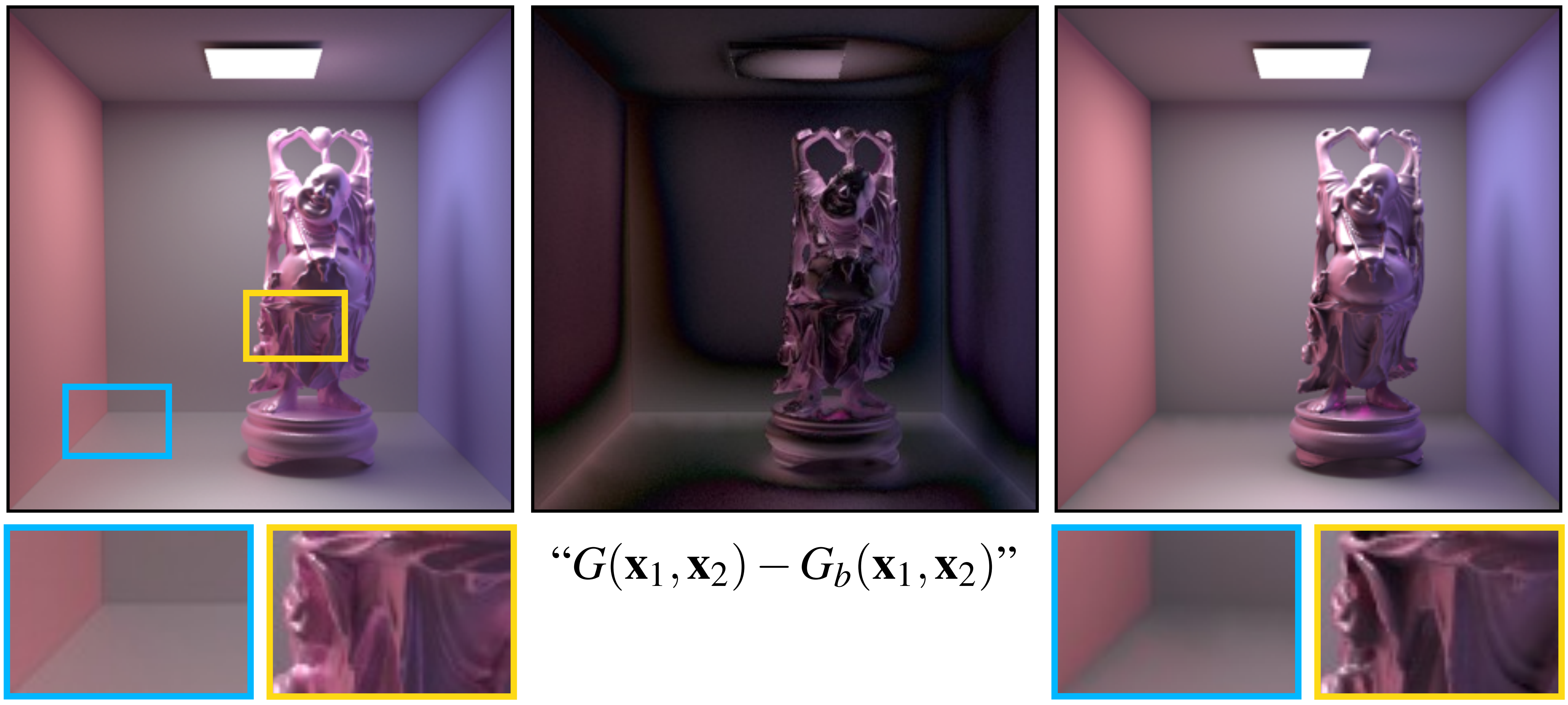
using $G_b(\mathbf{x}_1,\mathbf{x}_2)$

Here we have again the reference, the previous VPL rendering,... and on the right, a VPL rendering with the bounded geometry term.

You can see that it is darker around corners and cavities. And since this heavily degrades the quality of the rendered images, we shall now try to compensate for the missing energy and add it back.

Bounding & Compensation

Reference ← **Difference** → **VPLs with bounded G**


“$G(\mathbf{x}_1, \mathbf{x}_2) - G_b(\mathbf{x}_1, \mathbf{x}_2)$”

using $G_b(\mathbf{x}_1, \mathbf{x}_2)$

We need to compensate for the energy loss!

Here we have again the reference, the previous VPL rendering,... and on the right, a VPL rendering with the bounded geometry term.

You can see that it is darker around corners and cavities. And since this heavily degrades the quality of the rendered images, we shall now try to compensate for the missing energy and add it back.

Bounding & Compensation

Expressing the energy loss

We first need to express the missing energy and for that I will have to use few equations unfortunately, but it should be fairly easy to follow.

First we define a general light transport operator T.

If you are not familiar with the operator notation,... thing of T as an operation that takes the outgoing radiance in the scene.... and adds one bounce to it.

Now we can modify T a bit by bounding the geometry term, and this we will call the bounded transport operator Tb.

And we can also express the energy that has been removed by bounding, and this yields the residual transport operator Tr.

You can now forget about these equations ,... all you need to keep in mind is that the original transport operator is basically a sum of the bounded and the residual operators.

Bounding & Compensation

Expressing the energy loss

- light transport operator $\mathbf{T}$:

$$(\mathbf{T}L)(\mathbf{x}_1 \to \mathbf{x}_0) = \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2)\, V(\mathbf{x}_1, \mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

We first need to express the missing energy and for that I will have to use few equations unfortunately, but it should be fairly easy to follow.

First we define a general light transport operator T.

If you are not familiar with the operator notation,... thing of T as an operation that takes the outgoing radiance in the scene.... and adds one bounce to it.

Now we can modify T a bit by bounding the geometry term, and this we will call the bounded transport operator Tb.

And we can also express the energy that has been removed by bounding, and this yields the residual transport operator Tr.

You can now forget about these equations ,... all you need to keep in mind is that the original transport operator is basically a sum of the bounded and the residual operators.

Bounding & Compensation

Expressing the energy loss

- light transport operator $\mathbf{T}$:

$$(\mathbf{T}L)(\mathbf{x}_1 \to \mathbf{x}_0) = \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2)\, V(\mathbf{x}_1, \mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

- **bounded** light transport operator $\mathbf{T_b}$:

$$(\mathbf{T_b}L)(\mathbf{x}_1 \to \mathbf{x}_0) = \int_A f(\mathbf{x}_1)\, \min(G(\mathbf{x}_1, \mathbf{x}_2), b)\, V(\mathbf{x}_1, \mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

We first need to express the missing energy and for that I will have to use few equations unfortunately, but it should be fairly easy to follow.

First we define a general light transport operator T.

If you are not familiar with the operator notation,... thing of T as an operation that takes the outgoing radiance in the scene.... and adds one bounce to it.

Now we can modify T a bit by bounding the geometry term, and this we will call the bounded transport operator Tb.

And we can also express the energy that has been removed by bounding, and this yields the residual transport operator Tr.

You can now forget about these equations ,... all you need to keep in mind is that the original transport operator is basically a sum of the bounded and the residual operators.

Bounding & Compensation

Expressing the energy loss

- light transport operator $\mathbf{T}$:

$$(\mathbf{T}L)(\mathbf{x}_1 \to \mathbf{x}_0) = \int_A f(\mathbf{x}_1)\, G(\mathbf{x}_1,\mathbf{x}_2)\, V(\mathbf{x}_1,\mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

- **bounded** light transport operator $\mathbf{T_b}$:

$$(\mathbf{T_b}L)(\mathbf{x}_1 \to \mathbf{x}_0) = \int_A f(\mathbf{x}_1)\, \min(G(\mathbf{x}_1,\mathbf{x}_2), b)\, V(\mathbf{x}_1,\mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

- **residual** light transport operator (compensation term) $\mathbf{T_r}$:

$$(\mathbf{T_r}L)(\mathbf{x}_1 \to \mathbf{x}_0) = \int_A f(\mathbf{x}_1)\, \max(G(\mathbf{x}_1,\mathbf{x}_2) - b, 0)\, V(\mathbf{x}_1,\mathbf{x}_2)\, L(\mathbf{x}_2 \to \mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)$$

We first need to express the missing energy and for that I will have to use few equations unfortunately, but it should be fairly easy to follow.

First we define a general light transport operator T.

If you are not familiar with the operator notation,... thing of T as an operation that takes the outgoing radiance in the scene.... and adds one bounce to it.

Now we can modify T a bit by bounding the geometry term, and this we will call the bounded transport operator Tb.

And we can also express the energy that has been removed by bounding, and this yields the residual transport operator Tr.

You can now forget about these equations ,... all you need to keep in mind is that the original transport operator is basically a sum of the bounded and the residual operators.

Bounding & Compensation

Expressing the energy loss

- light transport operator $\mathbf{T}$:

$$\mathbf{T}L = \mathbf{T}_{\mathbf{b}}L + \mathbf{T}_{\mathbf{r}}L$$

- **bounded** light transport operator $\mathbf{T}_{\mathbf{b}}$:

$$(\mathbf{T}_{\mathbf{b}}L)(\mathbf{x}_1\to\mathbf{x}_0) = \int_A f(\mathbf{x}_1)\,\mathrm{min}(G(\mathbf{x}_1,\mathbf{x}_2),b)\,V(\mathbf{x}_1,\mathbf{x}_2)\,L(\mathbf{x}_2\to\mathbf{x}_1)\,\mathrm{d}A(\mathbf{x}_2)$$

- **residual** light transport operator (compensation term) $\mathbf{T}_{\mathbf{r}}$:

$$(\mathbf{T}_{\mathbf{r}}L)(\mathbf{x}_1\to\mathbf{x}_0) = \int_A f(\mathbf{x}_1)\,\mathrm{max}(G(\mathbf{x}_1,\mathbf{x}_2)-b,0)\,V(\mathbf{x}_1,\mathbf{x}_2)\,L(\mathbf{x}_2\to\mathbf{x}_1)\,\mathrm{d}A(\mathbf{x}_2)$$

We first need to express the missing energy and for that I will have to use few equations unfortunately, but it should be fairly easy to follow.

First we define a general light transport operator T.

If you are not familiar with the operator notation,... thing of T as an operation that takes the outgoing radiance in the scene.... and adds one bounce to it.

Now we can modify T a bit by bounding the geometry term, and this we will call the bounded transport operator Tb.

And we can also express the energy that has been removed by bounding, and this yields the residual transport operator Tr.

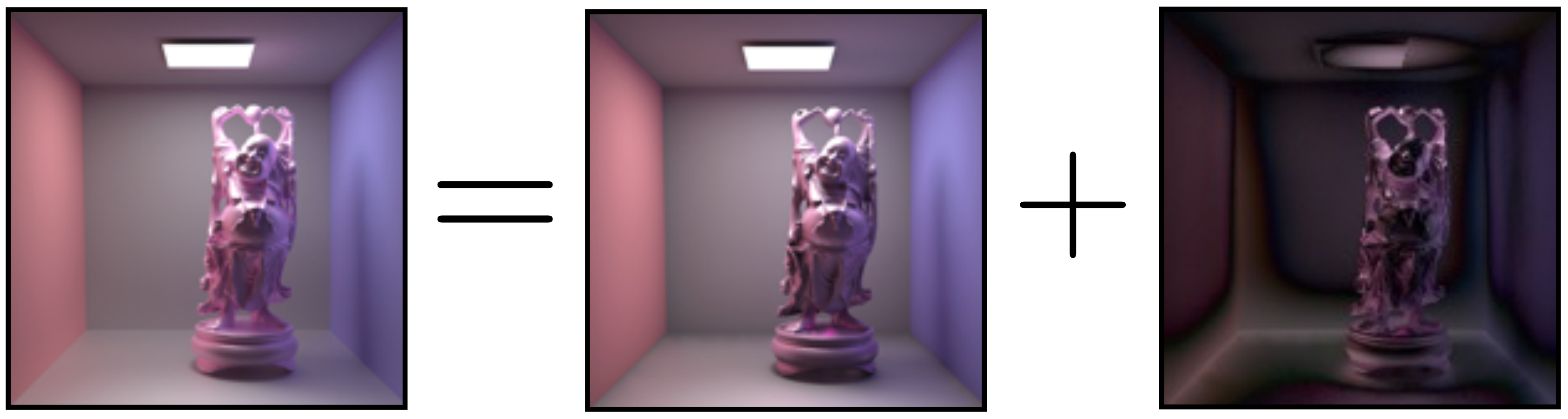
You can now forget about these equations ,... all you need to keep in mind is that the original transport operator is basically a sum of the bounded and the residual operators.

Bounding & Compensation

Expressing the energy loss

- light transport operator $\mathbf{T}$:

$$\mathbf{T}L = \mathbf{T}_{\mathbf{b}}L + \mathbf{T}_{\mathbf{r}}L$$



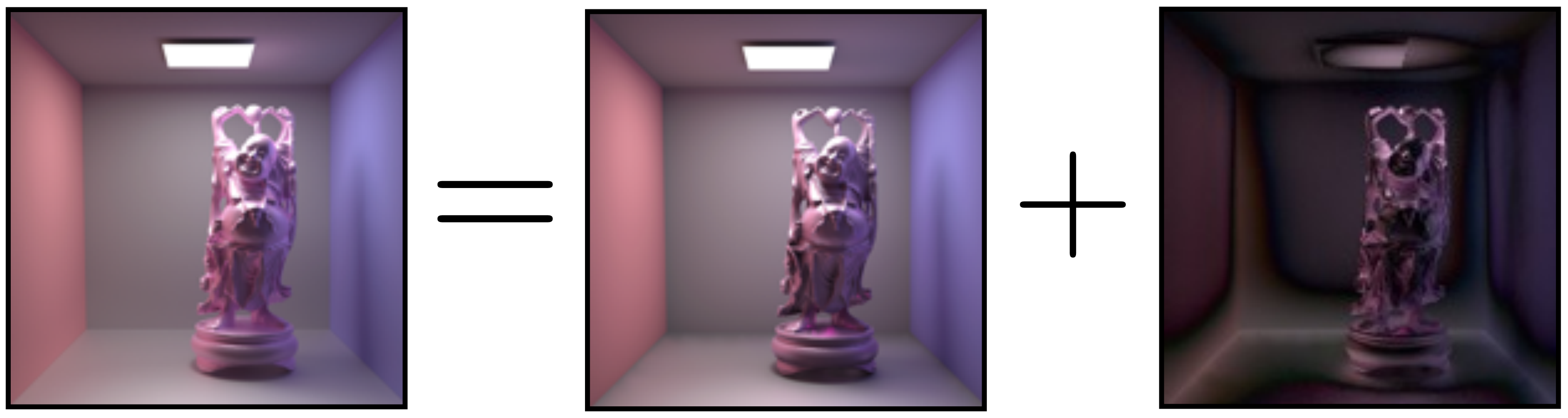
To compute all the light transport we will estimate the bounded transport using VPLs and the residual transport we need to try to estimate differently to avoid the splotches and I will now review few publications that propose to compute the residual transport slightly differently.

Bounding & Compensation

Expressing the energy loss

- light transport operator $\mathbf{T}$:

$$\mathbf{T}L = \mathbf{T}_{\mathbf{b}}L + \mathbf{T}_{\mathbf{r}}L$$



Estimate using VPLs

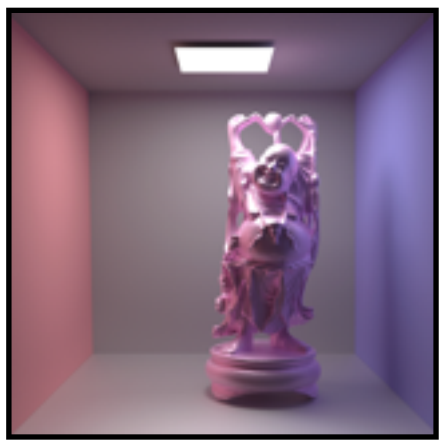
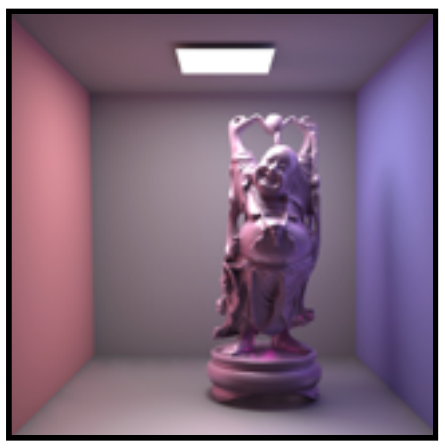
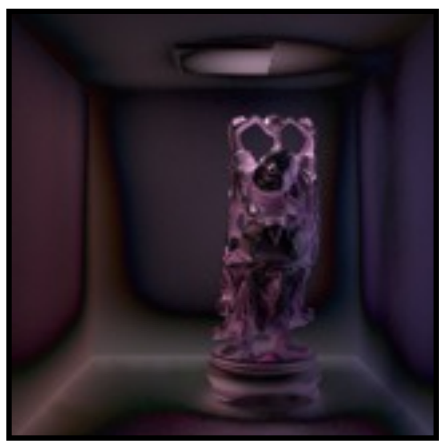
To compute all the light transport we will estimate the bounded transport using VPLs and the residual transport we need to try to estimate differently to avoid the splotches and I will now review few publications that propose to compute the residual transport slightly differently.

Bounding & Compensation

Expressing the energy loss

- light transport operator $\mathbf{T}$:

$$\mathbf{T}L = \mathbf{T}_{\mathbf{b}}L + \mathbf{T}_{\mathbf{r}}L$$


$=$

$+$


Estimate using VPLs

Estimate "differently"

To compute all the light transport we will estimate the bounded transport using VPLs and the residual transport we need to try to estimate differently to avoid the splotches and I will now review few publications that propose to compute the residual transport slightly differently.

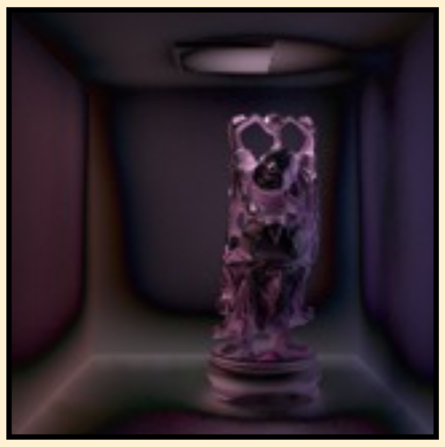
Bounding & Compensation

Expressing the energy loss

- light transport operator $\mathbf{T}$:

$$\mathbf{T}L = \mathbf{T_b}L + \mathbf{T_r}L$$

$=$

$+$


Estimate using VPLs

Estimate "differently"

[Kollig and Keller 2004]
[Raab et al. 2008]
[Davidovič et al. 2010]
[Novák et al. 2011]
[Engelhardt et al. 2012]

To compute all the light transport we will estimate the bounded transport using VPLs and the residual transport we need to try to estimate differently to avoid the splotches and I will now review few publications that propose to compute the residual transport slightly differently.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)

The first publication addressing the bias issue was the one by Kollig and Keller called “Global Illumination in the presence of weak singularities” from 2004.

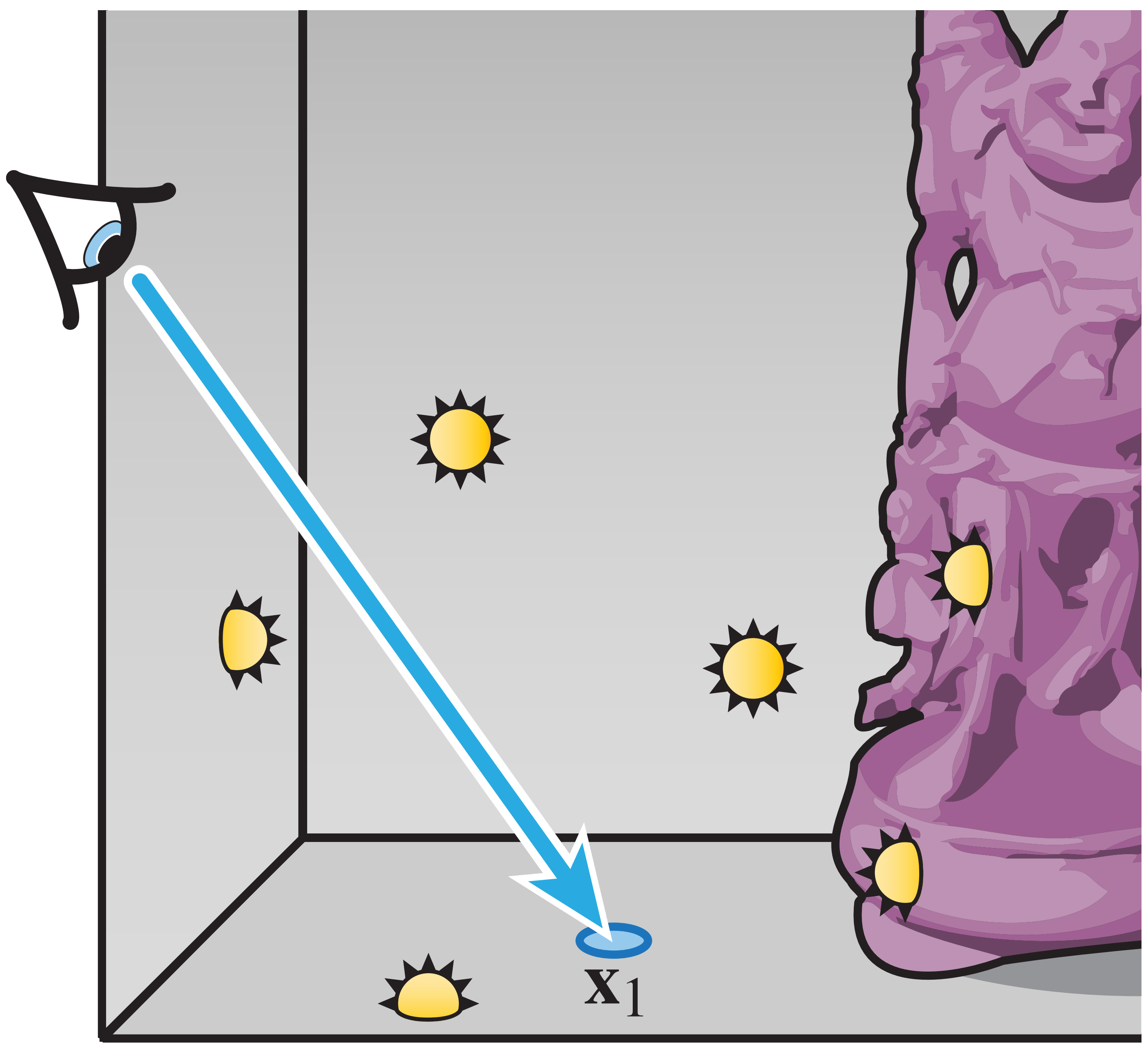
They define a so-called bias compensation term, which recovers the residual transport.

The algorithm works like this...

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



The first publication addressing the bias issue was the one by Kollig and Keller called “Global Illumination in the presence of weak singularities” from 2004.

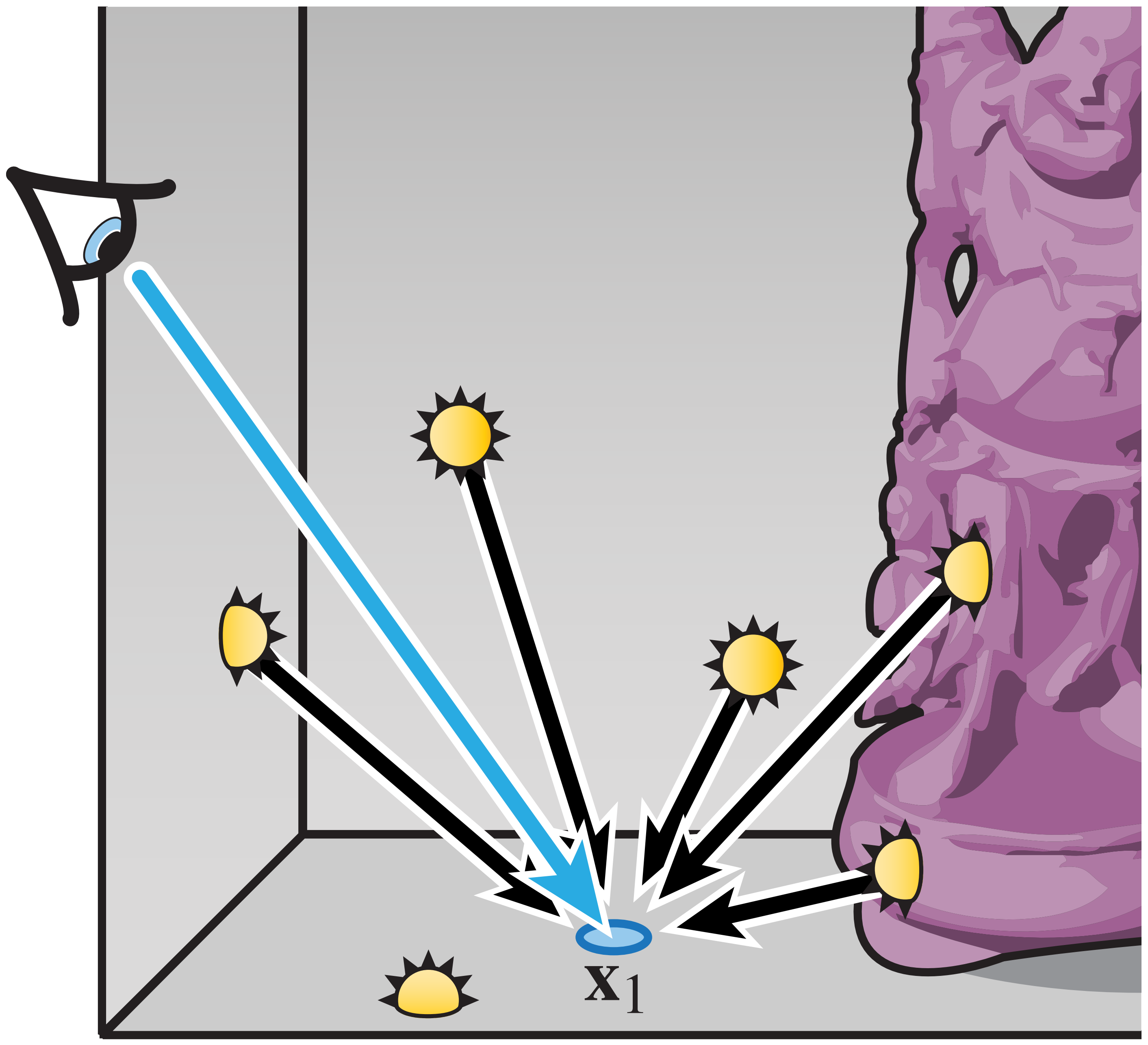
They define a so-called bias compensation term, which recovers the residual transport.

The algorithm works like this...

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



They first estimate the illumination of the shading point using the bounded operator, which will clamp contributions from VPLs on nearby surfaces.

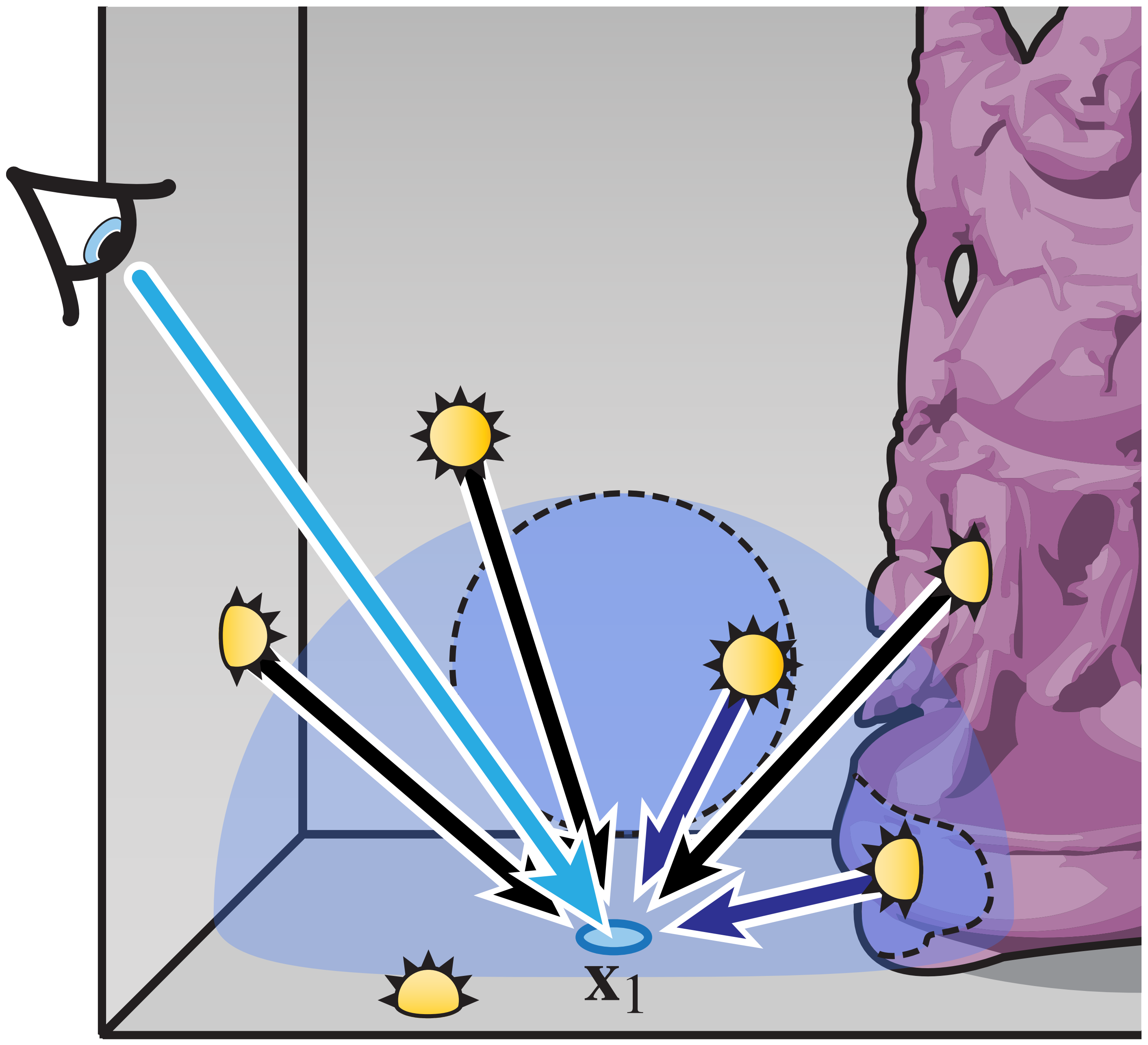
Here you can see the region, where the bounding occurs. Then they propose to add the clamped energy by shooting a ray and if this ray happens to hit a surface inside the bounding region, such as in this case,
they compute the illumination of this point and transport the light to x1 using the residual transport operator.

And since bounding occurs also at the red compensation vertex, they need to repeat the compensation recursively until the compensation vertex is outside the bounding region, such as this one.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



They first estimate the illumination of the shading point using the bounded operator, which will clamp contributions from VPLs on nearby surfaces.

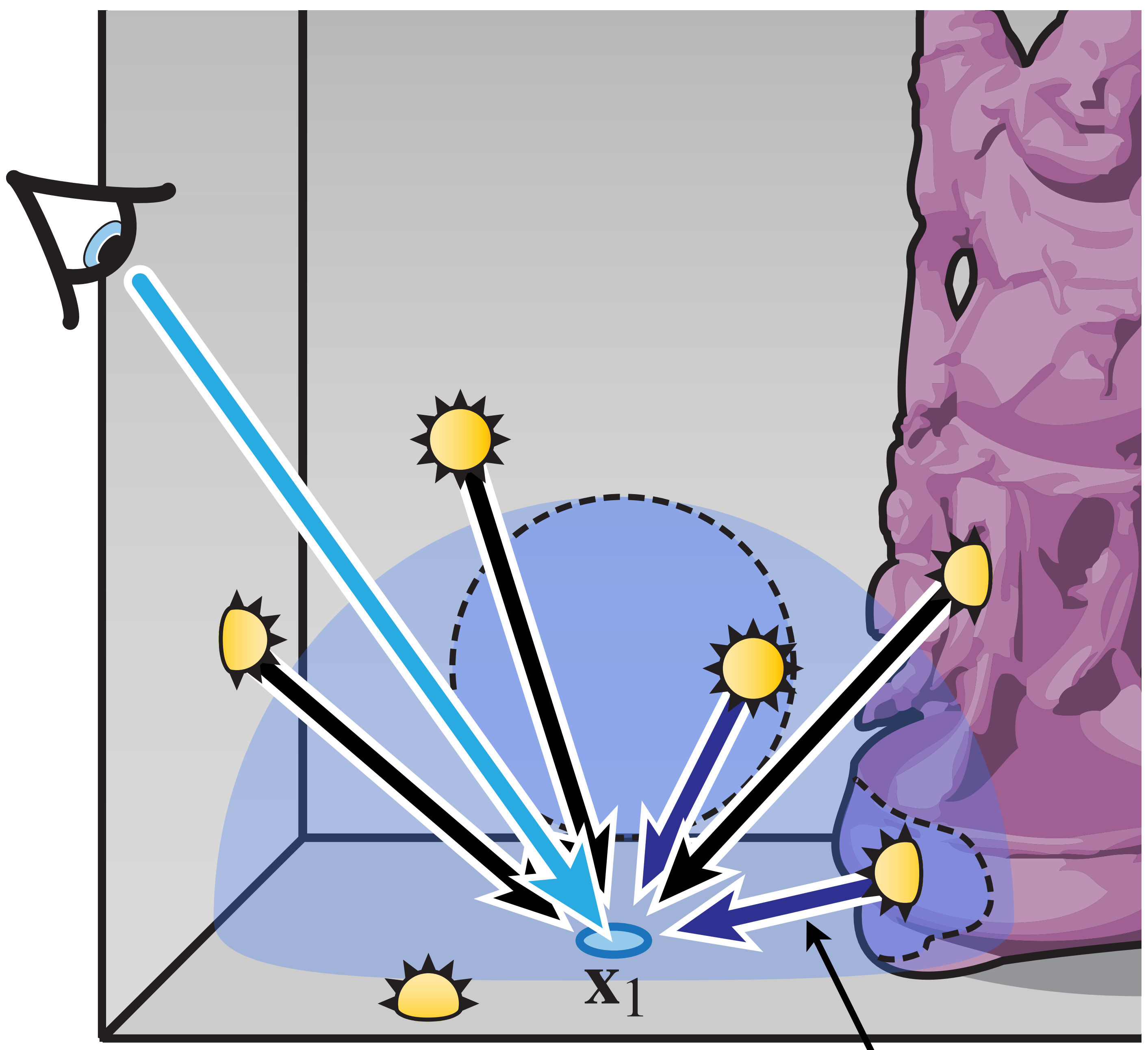
Here you can see the region, where the bounding occurs. Then they propose to add the clamped energy by shooting a ray and if this ray happens to hit a surface inside the bounding region, such as in this case,
they compute the illumination of this point and transport the light to x1 using the residual transport operator.

And since bounding occurs also at the red compensation vertex, they need to repeat the compensation recursively until the compensation vertex is outside the bounding region, such as this one.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



They first estimate the illumination of the shading point using the bounded operator, which will clamp contributions from VPLs on nearby surfaces.

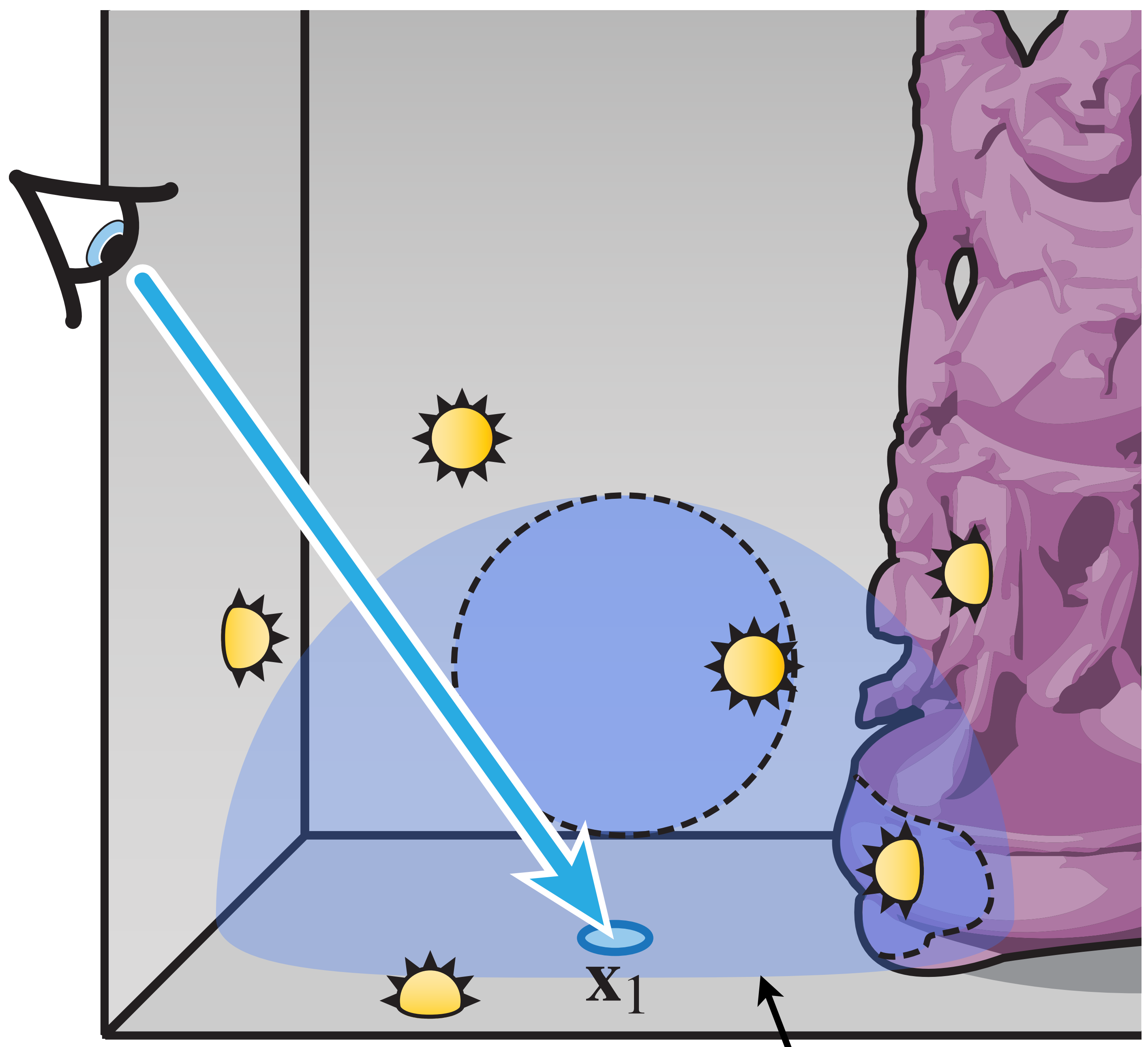
Here you can see the region, where the bounding occurs. Then they propose to add the clamped energy by shooting a ray and if this ray happens to hit a surface inside the bounding region, such as in this case,
they compute the illumination of this point and transport the light to x1 using the residual transport operator.

And since bounding occurs also at the red compensation vertex, they need to repeat the compensation recursively until the compensation vertex is outside the bounding region, such as this one.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



They first estimate the illumination of the shading point using the bounded operator, which will clamp contributions from VPLs on nearby surfaces.

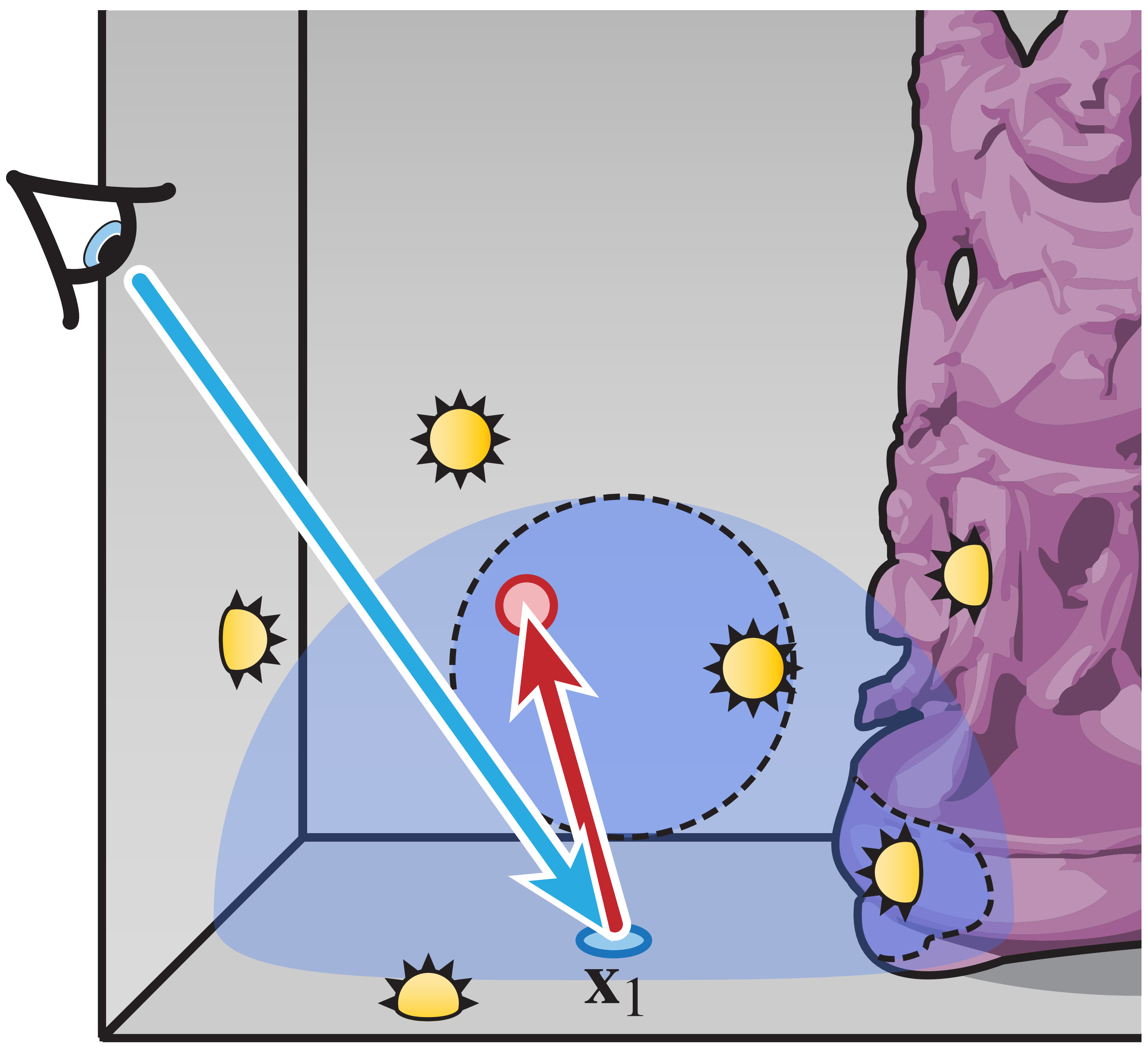
Here you can see the region, where the bounding occurs. Then they propose to add the clamped energy by shooting a ray and if this ray happens to hit a surface inside the bounding region, such as in this case,
they compute the illumination of this point and transport the light to x1 using the residual transport operator.

And since bounding occurs also at the red compensation vertex, they need to repeat the compensation recursively until the compensation vertex is outside the bounding region, such as this one.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



They first estimate the illumination of the shading point using the bounded operator, which will clamp contributions from VPLs on nearby surfaces.

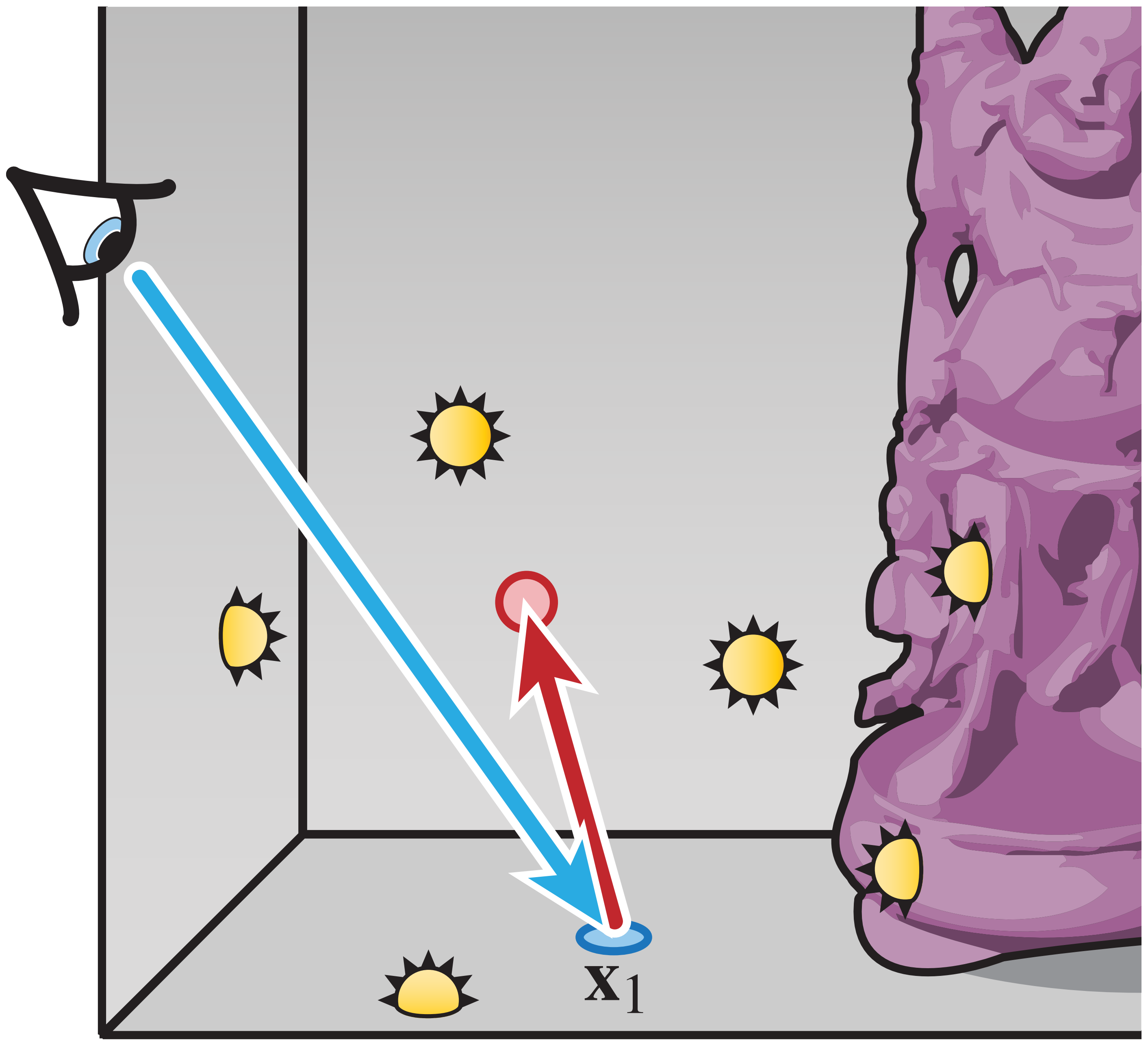
Here you can see the region, where the bounding occurs. Then they propose to add the clamped energy by shooting a ray and if this ray happens to hit a surface inside the bounding region, such as in this case,
they compute the illumination of this point and transport the light to x1 using the residual transport operator.

And since bounding occurs also at the red compensation vertex, they need to repeat the compensation recursively until the compensation vertex is outside the bounding region, such as this one.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



They first estimate the illumination of the shading point using the bounded operator, which will clamp contributions from VPLs on nearby surfaces.

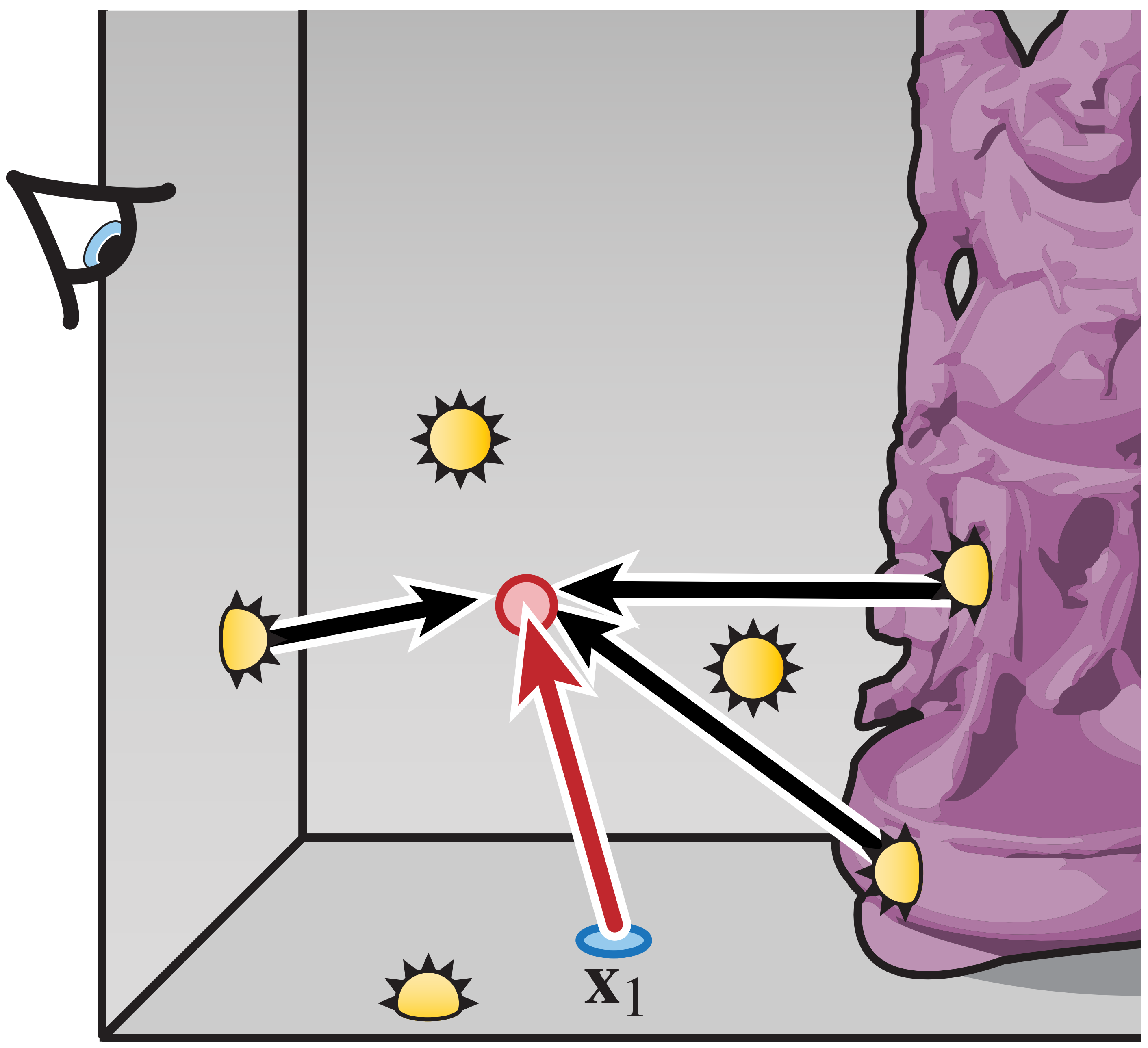
Here you can see the region, where the bounding occurs. Then they propose to add the clamped energy by shooting a ray and if this ray happens to hit a surface inside the bounding region, such as in this case,
they compute the illumination of this point and transport the light to x1 using the residual transport operator.

And since bounding occurs also at the red compensation vertex, they need to repeat the compensation recursively until the compensation vertex is outside the bounding region, such as this one.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



They first estimate the illumination of the shading point using the bounded operator, which will clamp contributions from VPLs on nearby surfaces.

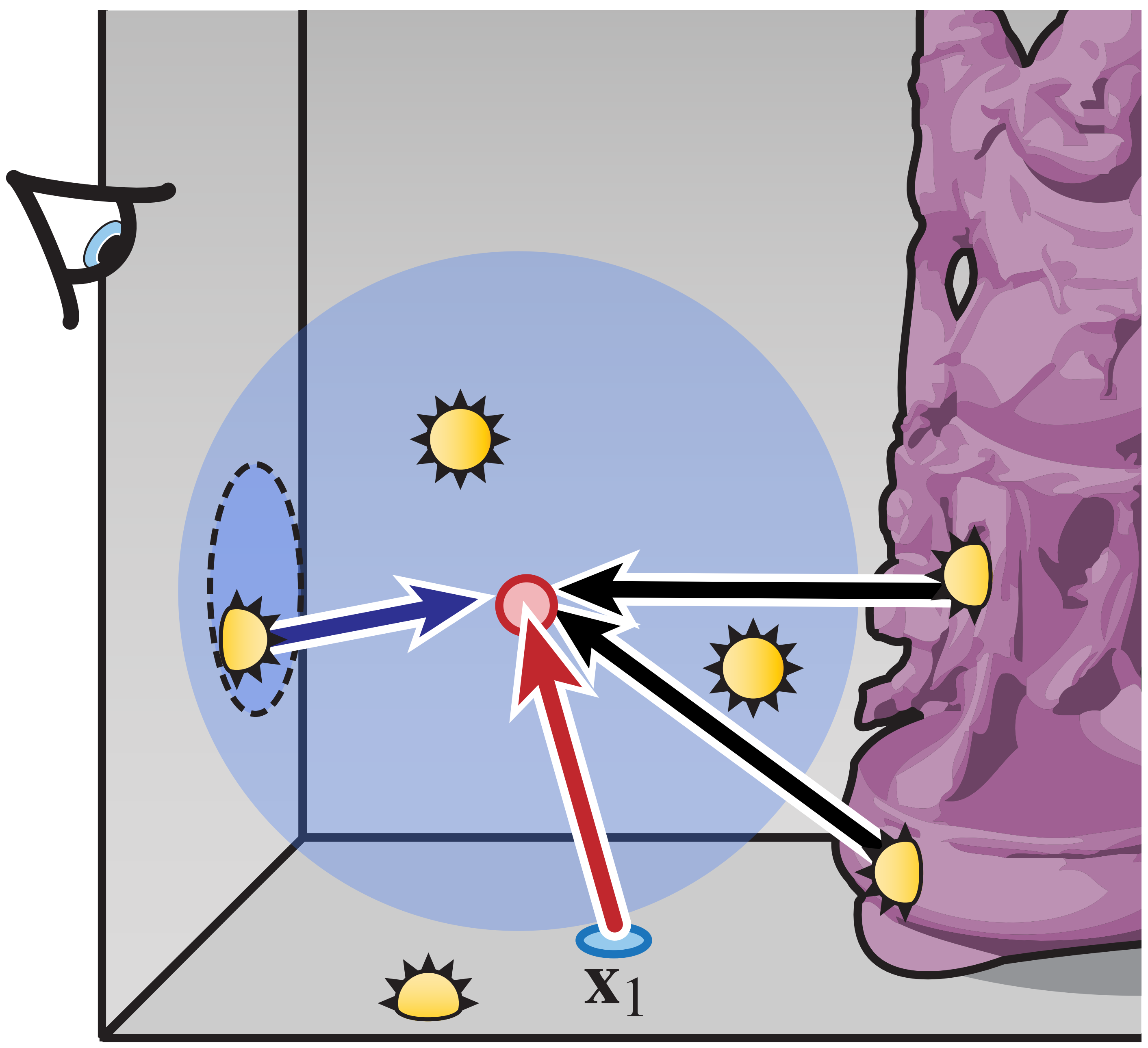
Here you can see the region, where the bounding occurs. Then they propose to add the clamped energy by shooting a ray and if this ray happens to hit a surface inside the bounding region, such as in this case,
they compute the illumination of this point and transport the light to x1 using the residual transport operator.

And since bounding occurs also at the red compensation vertex, they need to repeat the compensation recursively until the compensation vertex is outside the bounding region, such as this one.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



They first estimate the illumination of the shading point using the bounded operator, which will clamp contributions from VPLs on nearby surfaces.

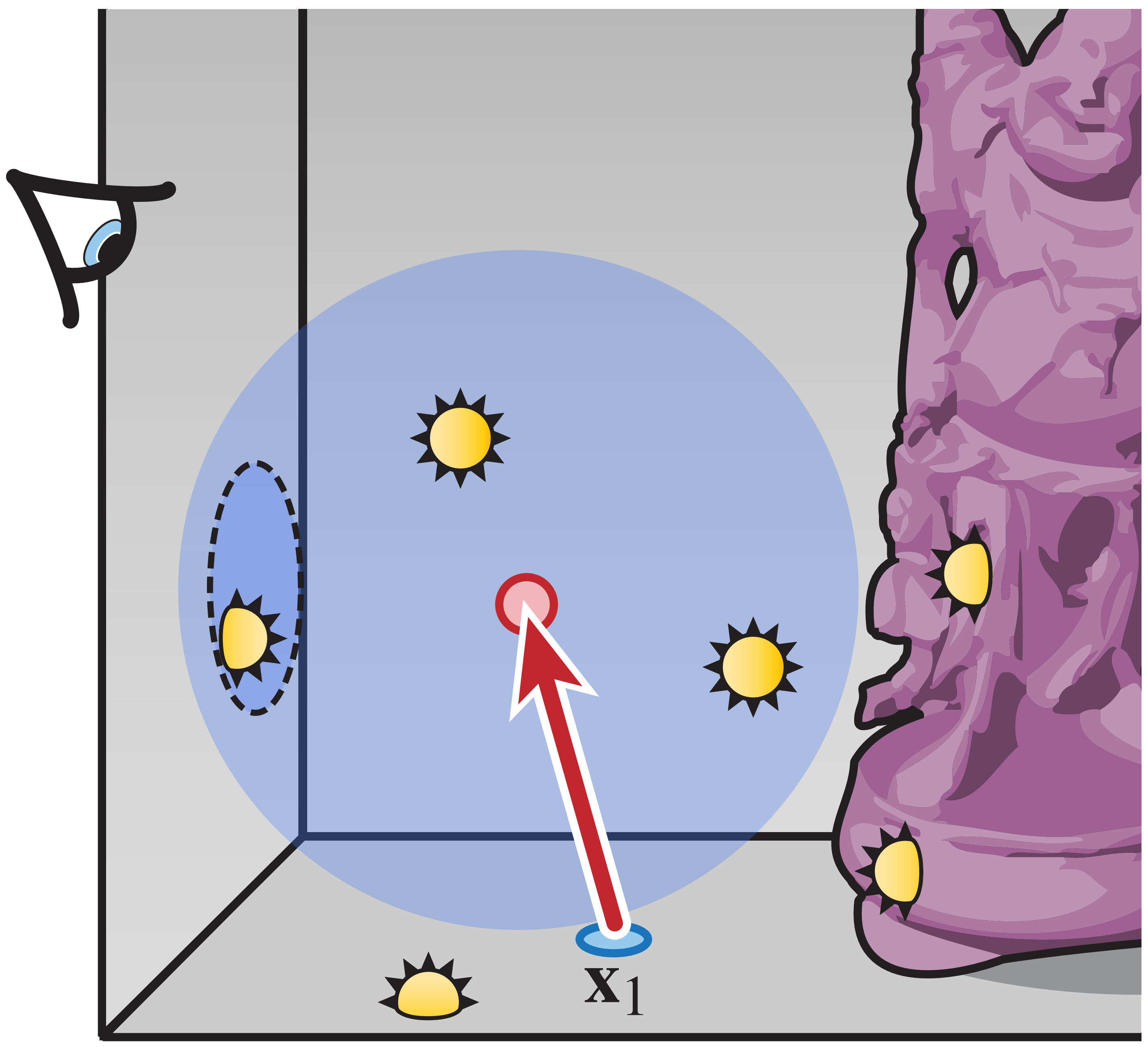
Here you can see the region, where the bounding occurs. Then they propose to add the clamped energy by shooting a ray and if this ray happens to hit a surface inside the bounding region, such as in this case,
they compute the illumination of this point and transport the light to x1 using the residual transport operator.

And since bounding occurs also at the red compensation vertex, they need to repeat the compensation recursively until the compensation vertex is outside the bounding region, such as this one.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



They first estimate the illumination of the shading point using the bounded operator, which will clamp contributions from VPLs on nearby surfaces.

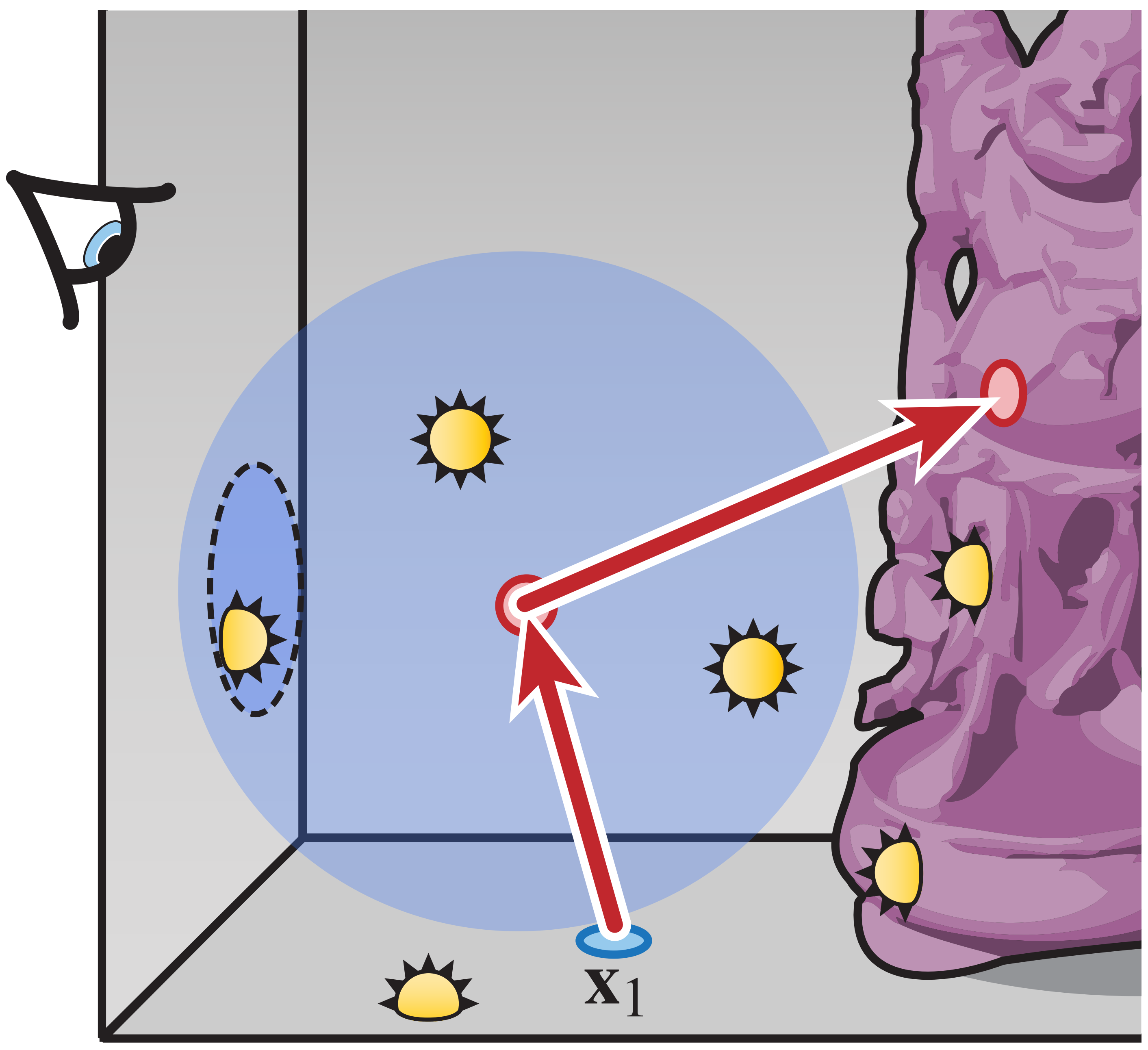
Here you can see the region, where the bounding occurs. Then they propose to add the clamped energy by shooting a ray and if this ray happens to hit a surface inside the bounding region, such as in this case,
they compute the illumination of this point and transport the light to x1 using the residual transport operator.

And since bounding occurs also at the red compensation vertex, they need to repeat the compensation recursively until the compensation vertex is outside the bounding region, such as this one.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)



They first estimate the illumination of the shading point using the bounded operator, which will clamp contributions from VPLs on nearby surfaces.

Here you can see the region, where the bounding occurs. Then they propose to add the clamped energy by shooting a ray and if this ray happens to hit a surface inside the bounding region, such as in this case,
they compute the illumination of this point and transport the light to x1 using the residual transport operator.

And since bounding occurs also at the red compensation vertex, they need to repeat the compensation recursively until the compensation vertex is outside the bounding region, such as this one.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)

The advantage of this bias compensation technique is that it recovers all the energy that was removed by bounding,
which makes the algorithm unbiased again.

The biggest problem is that this technique is recursive; it simply degenerates to fairly expensive path tracing and the bias compensation can thus easily increase the rendering time by an order of magnitude.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)

Advantages:

- recovers all missing energy
- makes the algorithm unbiased

The advantage of this bias compensation technique is that it recovers all the energy that was removed by bounding,
which makes the algorithm unbiased again.

The biggest problem is that this technique is recursive; it simply degenerates to fairly expensive path tracing and the bias compensation can thus easily increase the rendering time by an order of magnitude.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]

- trace paths to compute the compensation term (residual transport)

Advantages:

- recovers all missing energy
- makes the algorithm unbiased

Disadvantages:

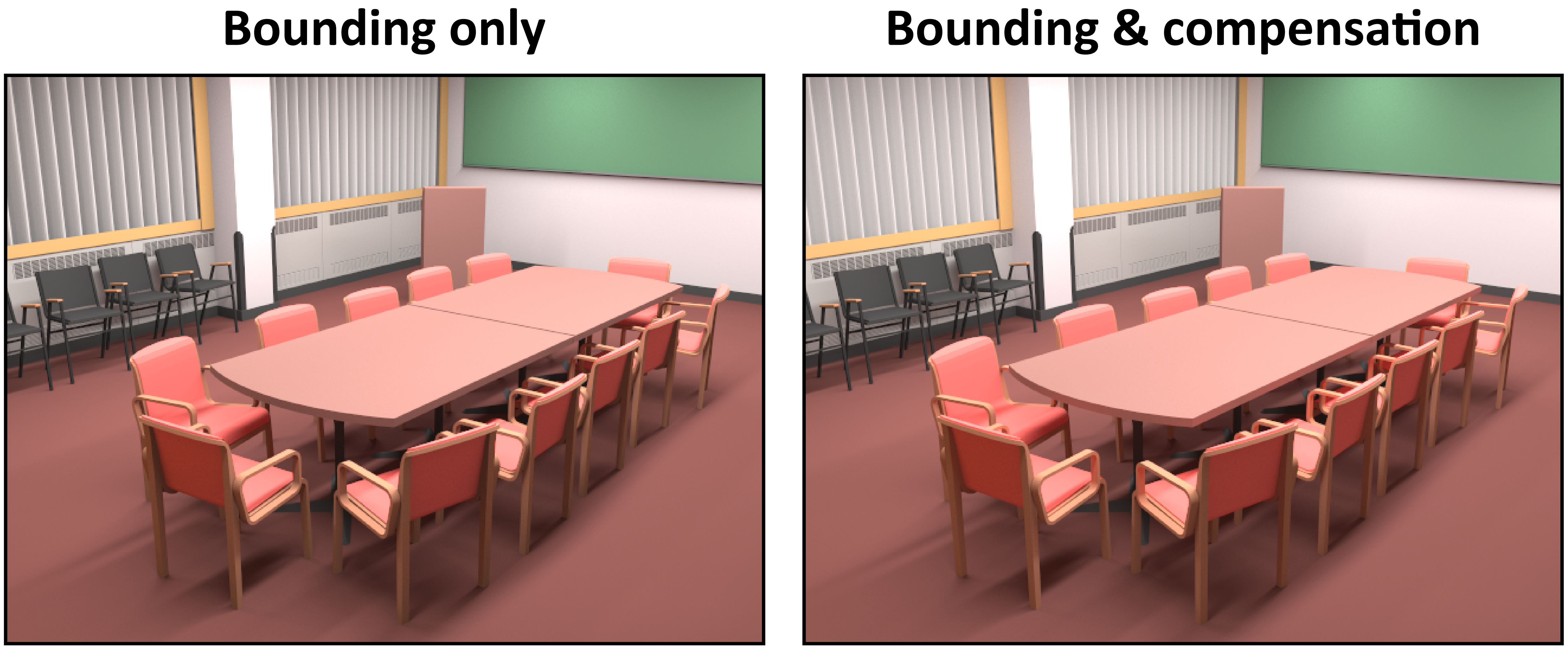
- recursive; degenerates to path tracing!
- very expensive
  - “recovering 10% of energy may take 90% of the rendering time”

The advantage of this bias compensation technique is that it recovers all the energy that was removed by bounding,
which makes the algorithm unbiased again.

The biggest problem is that this technique is recursive; it simply degenerates to fairly expensive path tracing and the bias compensation can thus easily increase the rendering time by an order of magnitude.

Bounding & Compensation

Bias compensation [Kollig and Keller 2004], [Raab et al. 2008]



Images courtesy of Kollig and Keller

Here we see a comparison, notice how the window blinds in the left biased image are much darker than in the unbiased rendering on the right.

Bounding & Compensation

In 2010, Davidovič and colleagues proposed a slightly different approach, which addresses the performance of the compensation.

Their idea is to, in addition to the global VPLs that distributed from the light source, also create local virtual lights that provide the residual transport.

Here we have the same situation as before, we have the compensation vertex, which gets illuminated by the global VPLs.

And to amortize the creation of this vertex the authors turn it into a local virtual point light that is used to illuminate several surface points around X1.

Bounding & Compensation

Local Virtual Lights [Davidovič et al. 2010]

- global (bounded) transport: VPLs
- local (residual) transport: on-demand local virtual lights

In 2010, Davidovič and colleagues proposed a slightly different approach, which addresses the performance of the compensation.

Their idea is to, in addition to the global VPLs that distributed from the light source, also create local virtual lights that provide the residual transport.

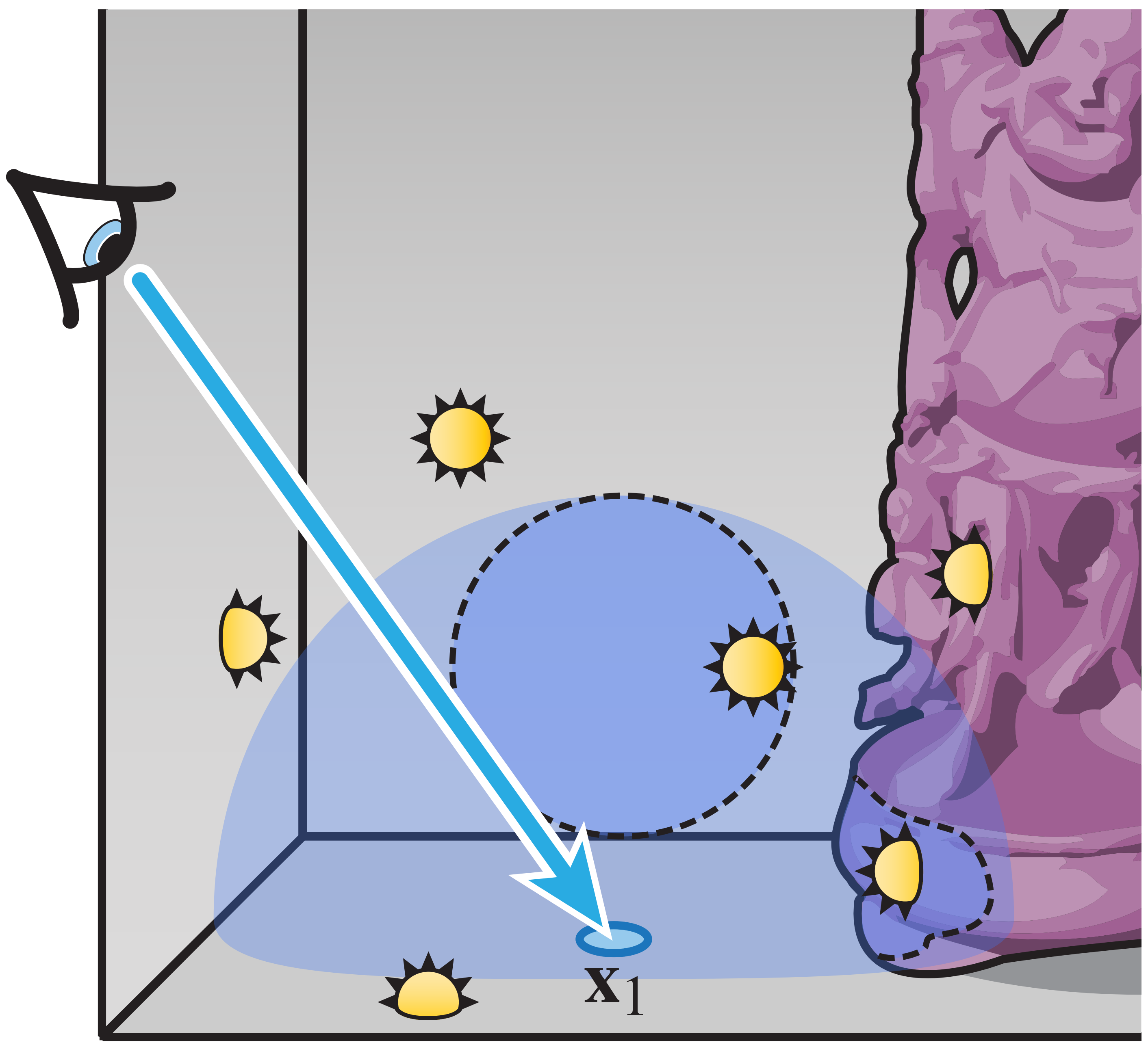
Here we have the same situation as before, we have the compensation vertex, which gets illuminated by the global VPLs.

And to amortize the creation of this vertex the authors turn it into a local virtual point light that is used to illuminate several surface points around X1.

Bounding & Compensation

Local Virtual Lights [Davidovič et al. 2010]

- global (bounded) transport: VPLs
- local (residual) transport: on-demand local virtual lights



In 2010, Davidovič and colleagues proposed a slightly different approach, which addresses the performance of the compensation.

Their idea is to, in addition to the global VPLs that distributed from the light source, also create local virtual lights that provide the residual transport.

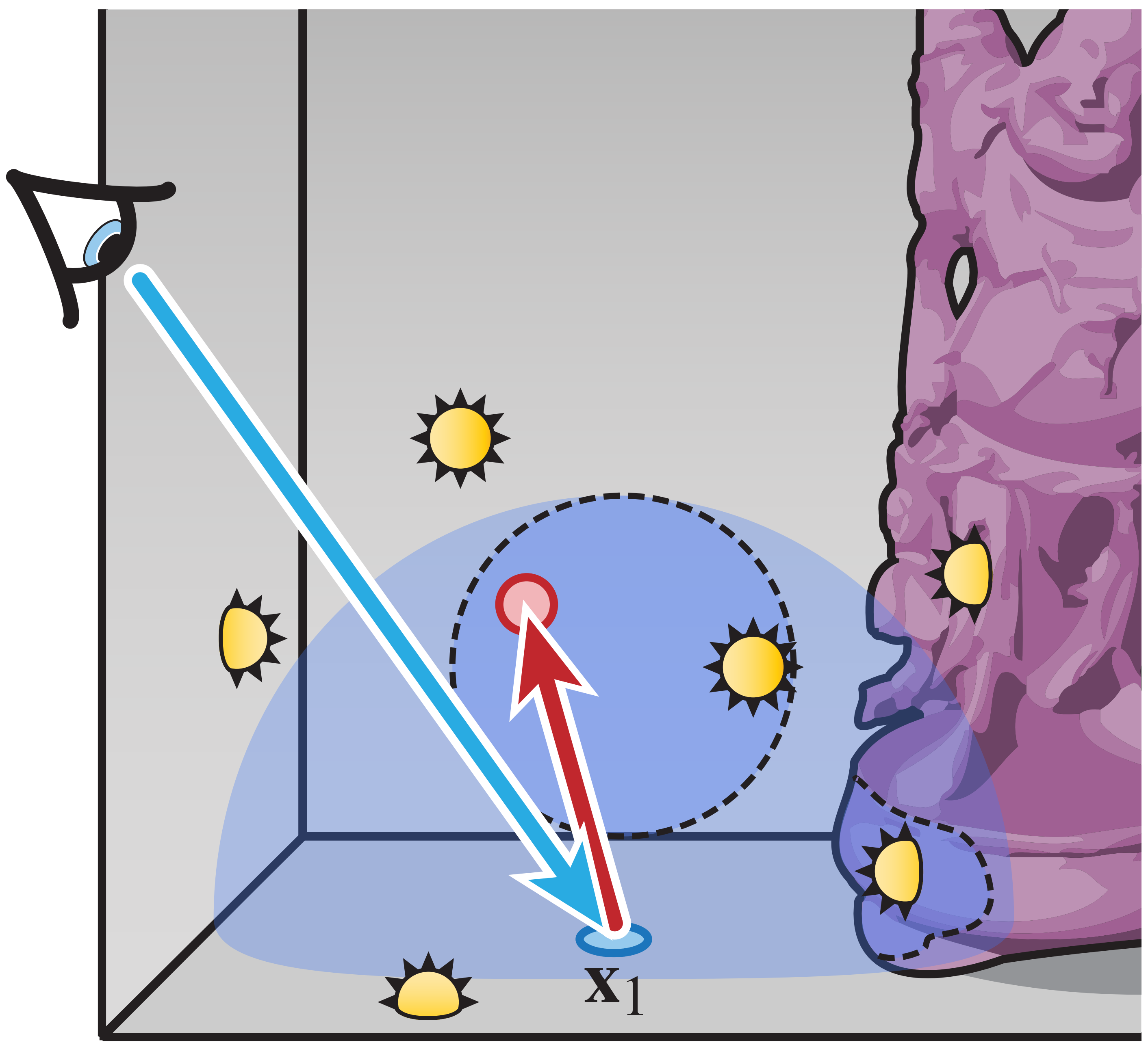
Here we have the same situation as before, we have the compensation vertex, which gets illuminated by the global VPLs.

And to amortize the creation of this vertex the authors turn it into a local virtual point light that is used to illuminate several surface points around X1.

Bounding & Compensation

Local Virtual Lights [Davidovič et al. 2010]

- global (bounded) transport: VPLs
- local (residual) transport: on-demand local virtual lights



In 2010, Davidovič and colleagues proposed a slightly different approach, which addresses the performance of the compensation.

Their idea is to, in addition to the global VPLs that distributed from the light source, also create local virtual lights that provide the residual transport.

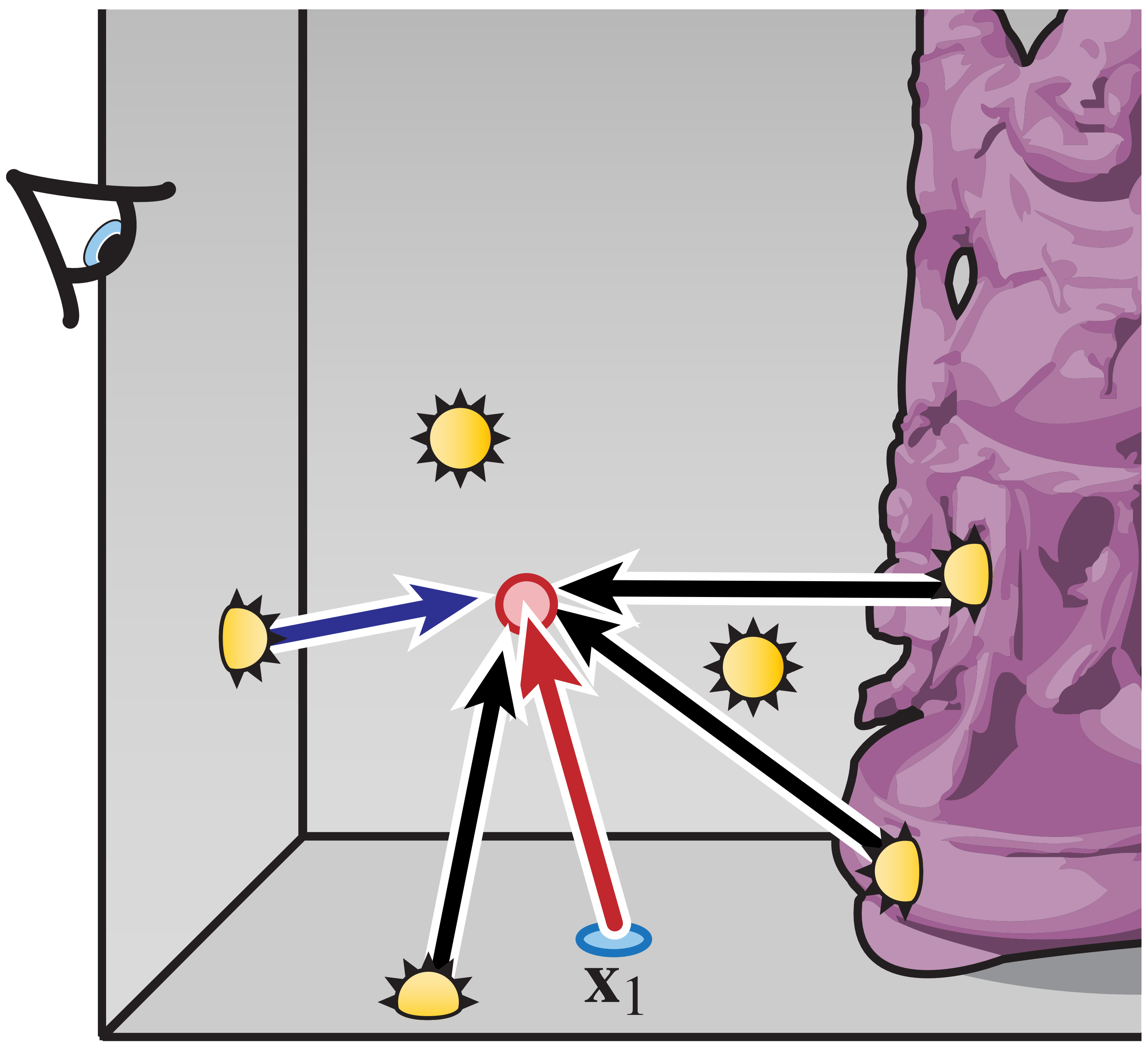
Here we have the same situation as before, we have the compensation vertex, which gets illuminated by the global VPLs.

And to amortize the creation of this vertex the authors turn it into a local virtual point light that is used to illuminate several surface points around X1.

Bounding & Compensation

Local Virtual Lights [Davidovič et al. 2010]

- global (bounded) transport: VPLs
- local (residual) transport: on-demand local virtual lights



In 2010, Davidovič and colleagues proposed a slightly different approach, which addresses the performance of the compensation.

Their idea is to, in addition to the global VPLs that distributed from the light source, also create local virtual lights that provide the residual transport.

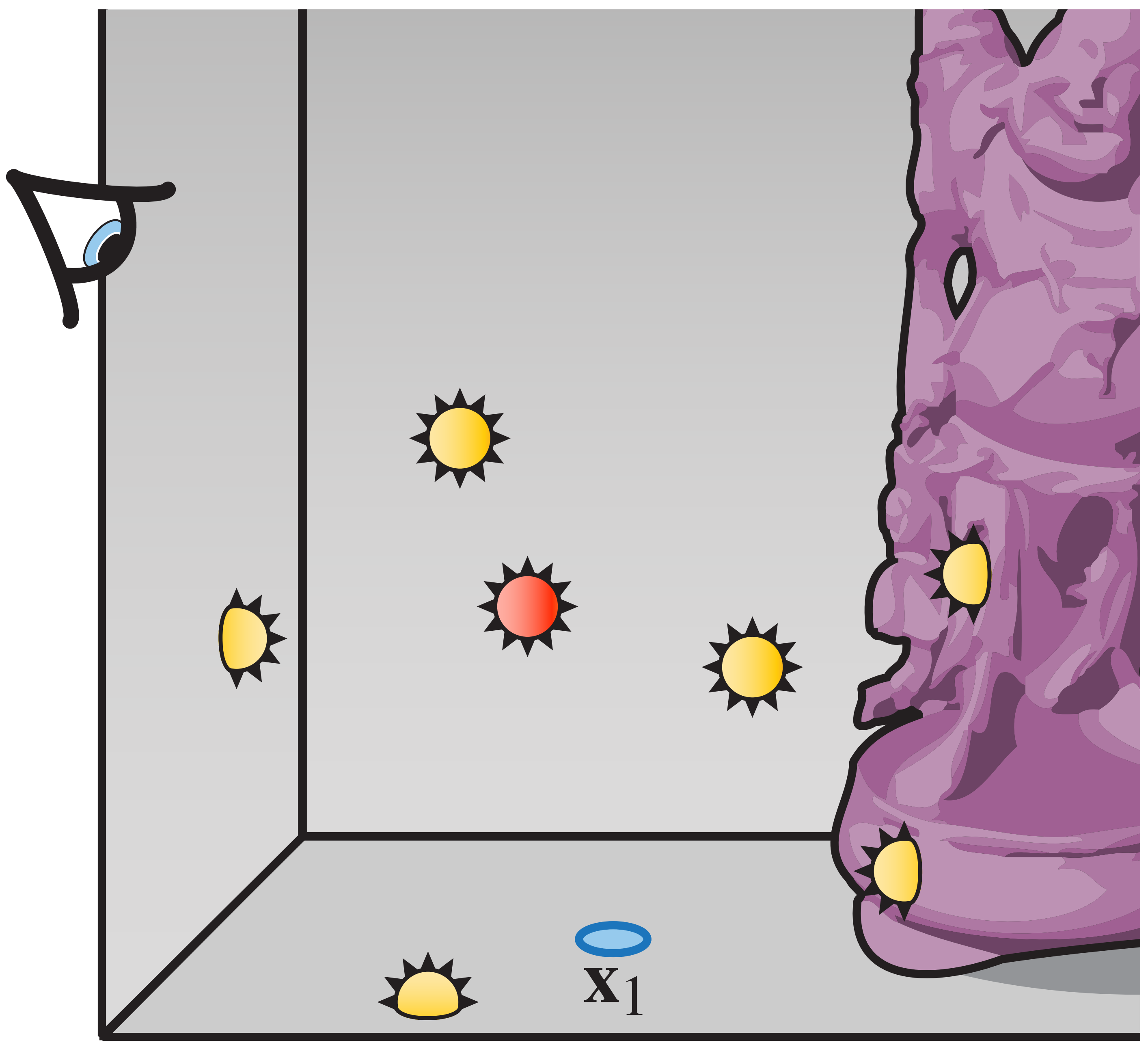
Here we have the same situation as before, we have the compensation vertex, which gets illuminated by the global VPLs.

And to amortize the creation of this vertex the authors turn it into a local virtual point light that is used to illuminate several surface points around X1.

Bounding & Compensation

Local Virtual Lights [Davidovič et al. 2010]

- global (bounded) transport: VPLs
- local (residual) transport: on-demand local virtual lights



In 2010, Davidovič and colleagues proposed a slightly different approach, which addresses the performance of the compensation.

Their idea is to, in addition to the global VPLs that distributed from the light source, also create local virtual lights that provide the residual transport.

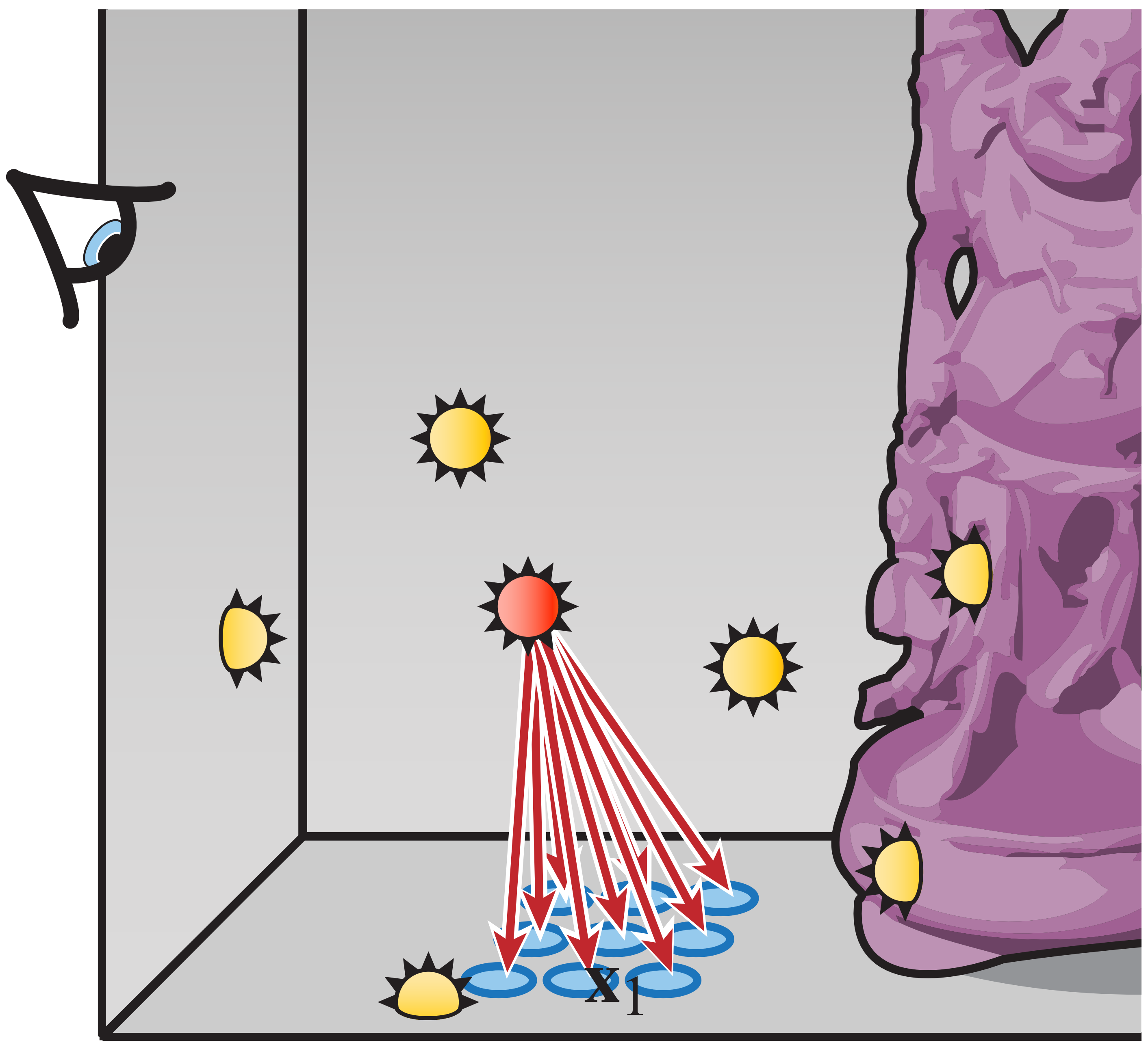
Here we have the same situation as before, we have the compensation vertex, which gets illuminated by the global VPLs.

And to amortize the creation of this vertex the authors turn it into a local virtual point light that is used to illuminate several surface points around X1.

Bounding & Compensation

Local Virtual Lights [Davidovič et al. 2010]

- global (bounded) transport: VPLs
- local (residual) transport: on-demand local virtual lights



In 2010, Davidovič and colleagues proposed a slightly different approach, which addresses the performance of the compensation.

Their idea is to, in addition to the global VPLs that distributed from the light source, also create local virtual lights that provide the residual transport.

Here we have the same situation as before, we have the compensation vertex, which gets illuminated by the global VPLs.

And to amortize the creation of this vertex the authors turn it into a local virtual point light that is used to illuminate several surface points around X1.

Bounding & Compensation

Local Virtual Lights [Davidovič et al. 2010]

- global (bounded) transport: VPLs
- local (residual) transport: on-demand local virtual lights

When compared to the original bias compensation, this technique is more efficient since it uses each compensation vertex for multiple pixels.
It also handles well the glossy inter-reflections.

One drawback is that the implementation approximates visibility, which can lead to missing indirect shadows.

Bounding & Compensation

Local Virtual Lights [Davidovič et al. 2010]

- global (bounded) transport: VPLs
- local (residual) transport: on-demand local virtual lights

Advantages:

- faster than Kollig and Keller's approach
- amortizes creation of each local light over several pixels
- handles glossy inter-reflection

When compared to the original bias compensation, this technique is more efficient since it uses each compensation vertex for multiple pixels.
It also handles well the glossy inter-reflections.

One drawback is that the implementation approximates visibility, which can lead to missing indirect shadows.

Bounding & Compensation

Local Virtual Lights [Davidovič et al. 2010]

- global (bounded) transport: VPLs
- local (residual) transport: on-demand local virtual lights

Advantages:

- faster than Kollig and Keller's approach
- amortizes creation of each local light over several pixels
- handles glossy inter-reflection

Disadvantages:

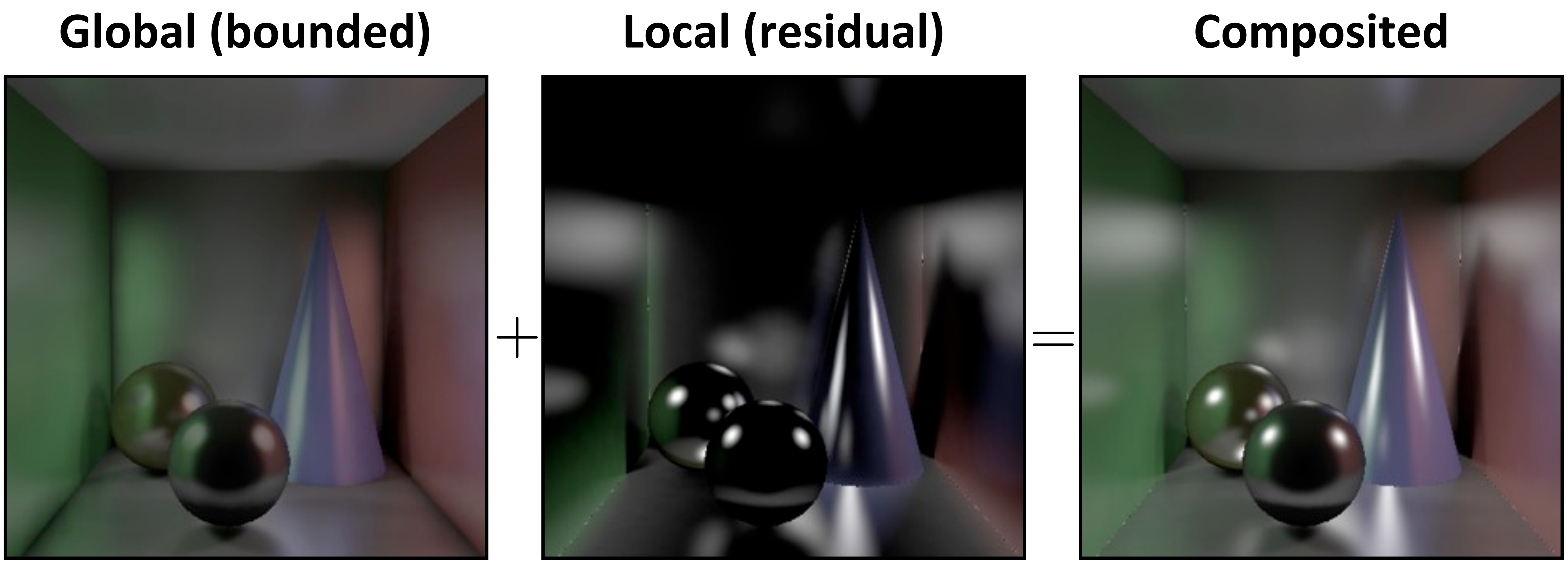
- proposed (involved) implementation approximates visibility

When compared to the original bias compensation, this technique is more efficient since it uses each compensation vertex for multiple pixels.
It also handles well the glossy inter-reflections.

One drawback is that the implementation approximates visibility, which can lead to missing indirect shadows.

Bounding & Compensation

Local Virtual Lights [Davidovič et al. 2010]



Images courtesy of Davidovič et a.

Here we see the two components, the global bounded and the local residual transport and the final composited image.

Bounding & Compensation

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process

- Rendering Equation:

$$L = L_{\mathrm{e}} + \mathbf{T}L$$

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process

- Rendering Equation:

$$\begin{aligned} L &= L_\mathrm{e} + \mathbf{T}L \\ &\approx L_\mathrm{e} + \mathbf{T}L_\mathrm{e} + \mathbf{T}\hat{L} \end{aligned}$$

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process

- Rendering Equation:

$$\begin{aligned} L &= L_{\mathrm{e}} + \mathbf{T}L \\ &\approx \underbrace{L_{\mathrm{e}}}_{\text{emission}} + \underbrace{\mathbf{T}L_{\mathrm{e}}}_{\text{direct illumination}} + \underbrace{\mathbf{T}\hat{L}}_{\text{indirect illumination computed using VPLs}} \end{aligned}$$

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process
- Rendering Equation:

$$\begin{aligned} L &= L_\mathrm{e} + \mathbf{T}L \\ &\approx L_\mathrm{e} + \mathbf{T}L_\mathrm{e} + \mathbf{T}\hat{L} \\ &\approx L_\mathrm{e} + \mathbf{T}L_\mathrm{e} + \mathbf{T}_{\mathbf{b}}\hat{L} + \mathbf{T}_{\mathbf{r}}\hat{L} \end{aligned}$$

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process
- Rendering Equation:

$$\begin{aligned} L &= L_e + \mathbf{T}L \\ &\approx L_e + \mathbf{T}L_e + \mathbf{T}\hat{L} \\ &\approx L_e + \mathbf{T}L_e + \underbrace{\mathbf{T}_{\mathbf{b}}}_{\text{bounded}}\hat{L} + \underbrace{\mathbf{T}_{\mathbf{r}}}_{\text{residual}}\hat{L} \end{aligned}$$

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process
- Rendering Equation:

$$\begin{aligned} L &= L_e + \mathbf{T}L \\ &\approx L_e + \mathbf{T}L_e + \mathbf{T}\hat{L} \\ &\approx L_e + \mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L} + \mathbf{T}_{\mathbf{r}}\hat{L} \\ &\approx L_e + \mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L} + \mathbf{T}_{\mathbf{r}}(L - L_e) \end{aligned}$$

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process
- Rendering Equation:

$$\begin{aligned} L &= L_{\mathrm{e}} + \mathbf{T}L \\ &\approx L_{\mathrm{e}} + \mathbf{T}L_{\mathrm{e}} + \mathbf{T}\hat{L} \\ &\approx L_{\mathrm{e}} + \mathbf{T}L_{\mathrm{e}} + \mathbf{T}_{\mathbf{b}}\hat{L} + \mathbf{T}_{\mathbf{r}}\hat{L} \\ &\approx L_{\mathrm{e}} + \mathbf{T}L_{\mathrm{e}} + \mathbf{T}_{\mathbf{b}}\hat{L} + \mathbf{T}_{\mathbf{r}}(\underbrace{L - L_{\mathrm{e}}}_{\text{recursively expand}}) \end{aligned}$$

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process

- Rendering Equation:

$$\begin{aligned} L &= L_e + \mathbf{T}L \\ &\approx L_e + \mathbf{T}L_e + \mathbf{T}\hat{L} \\ &\approx L_e + \mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L} + \mathbf{T}_{\mathbf{r}}\hat{L} \\ &\approx L_e + \mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L} + \mathbf{T}_{\mathbf{r}}(L - L_e) \\ &\approx L_e + \sum_{i=0}^{\infty} \mathbf{T}_{\mathbf{r}}^{i}(\mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L}) \end{aligned}$$

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process
- Rendering Equation:

$$\begin{aligned} L &= L_e + \mathbf{T}L \\ &\approx L_e + \mathbf{T}L_e + \mathbf{T}\hat{L} \\ &\approx L_e + \mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L} + \mathbf{T}_{\mathbf{r}}\hat{L} \\ &\approx L_e + \mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L} + \mathbf{T}_{\mathbf{r}}(L - L_e) \\ &\approx L_e + \sum_{i=0}^{\infty} \mathbf{T}_{\mathbf{r}}^{i} \underbrace{(\mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L})}_{\text{compute once and store}} \end{aligned}$$

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process
- Rendering Equation:

$$\begin{aligned} L &= L_\mathrm{e} + \mathbf{T}L \\ &\approx L_\mathrm{e} + \mathbf{T}L_\mathrm{e} + \mathbf{T}\hat{L} \\ &\approx L_\mathrm{e} + \mathbf{T}L_\mathrm{e} + \mathbf{T}_\mathbf{b}\hat{L} + \mathbf{T}_\mathbf{r}\hat{L} \\ &\approx L_\mathrm{e} + \mathbf{T}L_\mathrm{e} + \mathbf{T}_\mathbf{b}\hat{L} + \mathbf{T}_\mathbf{r}(L - L_\mathrm{e}) \\ &\approx L_\mathrm{e} + \underbrace{\sum_{i=0}^{\infty} \mathbf{T}_\mathbf{r}^i}_{\text{iteratively apply } \mathbf{T}_\mathbf{r}} \underbrace{(\mathbf{T}L_\mathrm{e} + \mathbf{T}_\mathbf{b}\hat{L})}_{\text{compute once and store}} \end{aligned}$$

Carsten, Thomas Engelhardt and me, we also had a look on how to compensate for the bias efficiently, preferably in realtime.

We found out that we can approximate the residual transport using post-processing filter in screen-space. To convince you that this is possible I will now have to work out few equations, so bear with me for a moment.

Here we have the rendering equation, which we can expand once, to separate the emission, direct illumination, and indirect illumination that is computed using VPLs, and the last term we then further split into the bounded and residual transport.

And for the residual transport we should not use the VPLs, this would give us the splotches, so instead we plug in the corresponding general illumination.

By expanding the equation further we would notice that the rendering equation can be rewritten into this form...

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process
- Rendering Equation:

$$\approx L_\mathrm{e} + \sum_{i=0}^{\infty} \mathbf{T}_\mathbf{r}^{i} (\mathbf{T} L_\mathrm{e} + \mathbf{T}_\mathbf{b} \hat{L})$$

Here is one specific example:

We first render the direct and bounded indirect illumination.

And then we use this image as the input for the computing the residual transport in screen-space. The result of this is then added to the bounded solution and also used as the input for another screen-space integration step, which yields an extra bounce of the residual transport.

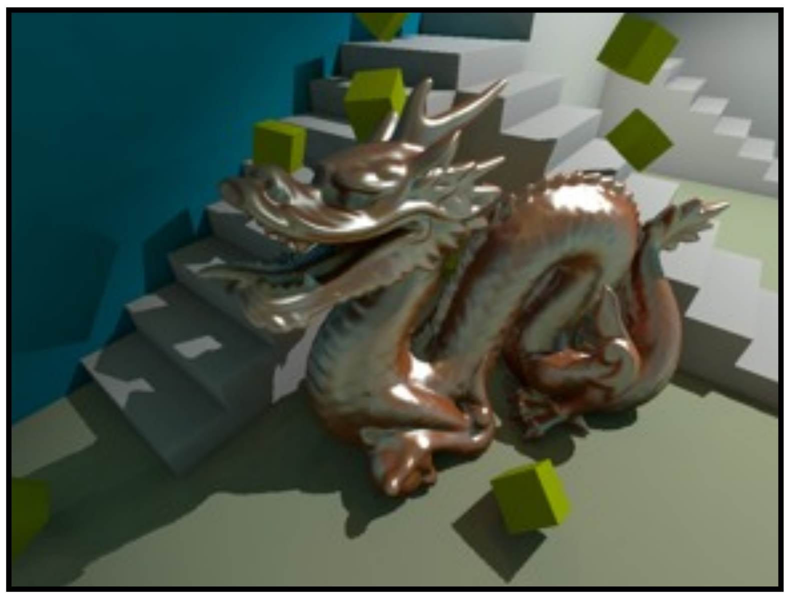
Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process
- Rendering Equation:

$$\approx L_\mathrm{e} + \sum_{i=0}^{\infty} \mathbf{T}_{\mathbf{r}}^{i} (\mathbf{T} L_\mathrm{e} + \mathbf{T}_{\mathbf{b}} \hat{L})$$

direct +
bounded indirect
illumination



$\mathbf{T}_{\mathbf{r}}^{0} (\mathbf{T} L_\mathrm{e} + \mathbf{T}_{\mathbf{b}} \hat{L})$

Here is one specific example:

We first render the direct and bounded indirect illumination.

And then we use this image as the input for the computing the residual transport in screen-space. The result of this is then added to the bounded solution and also used as the input for another screen-space integration step, which yields an extra bounce of the residual transport.

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

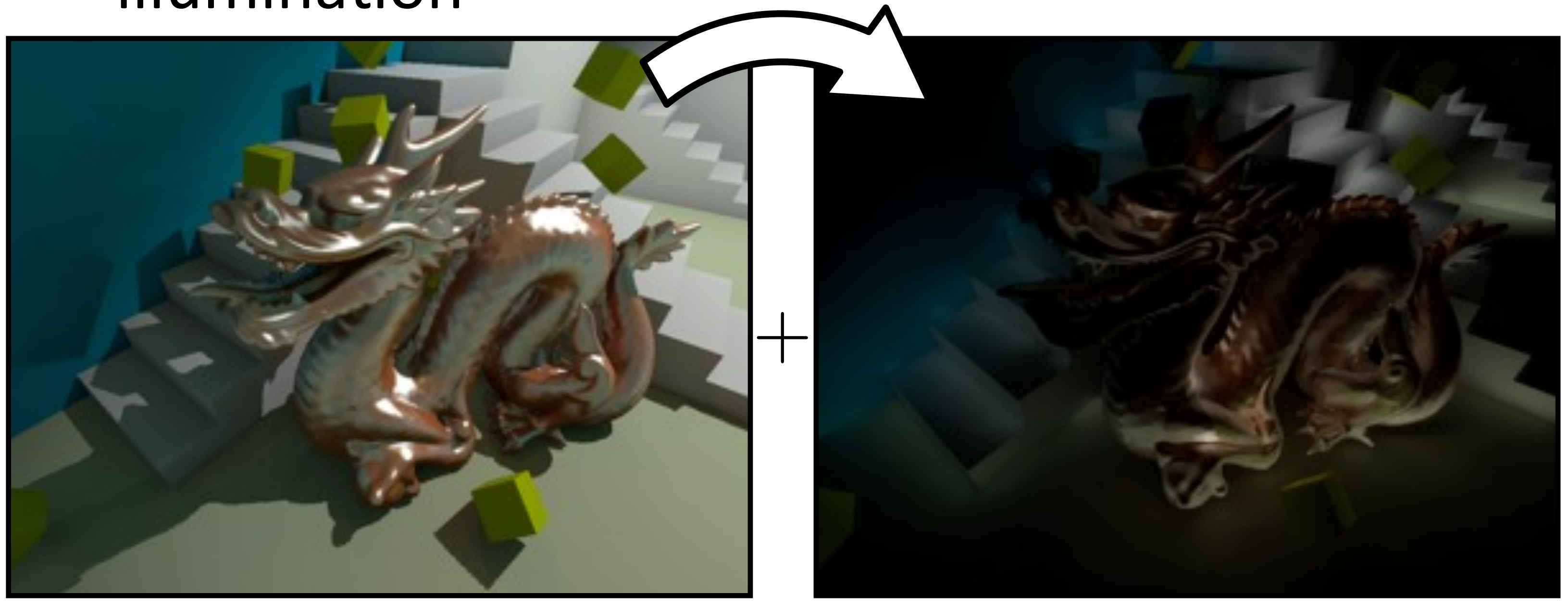
- residual transport is localized and can be applied in post-process

- Rendering Equation:

$$\approx L_e + \sum_{i=0}^{\infty} \mathbf{T}_{\mathbf{r}}^{i}(\mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L})$$

direct + bounded indirect illumination

residual transport in screen-space



$\mathbf{T}_{\mathbf{r}}^{0}(\mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L})$ $+$ $\mathbf{T}_{\mathbf{r}}^{1}(\mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L})$

Here is one specific example:

We first render the direct and bounded indirect illumination.

And then we use this image as the input for the computing the residual transport in screen-space. The result of this is then added to the bounded solution and also used as the input for another screen-space integration step, which yields an extra bounce of the residual transport.

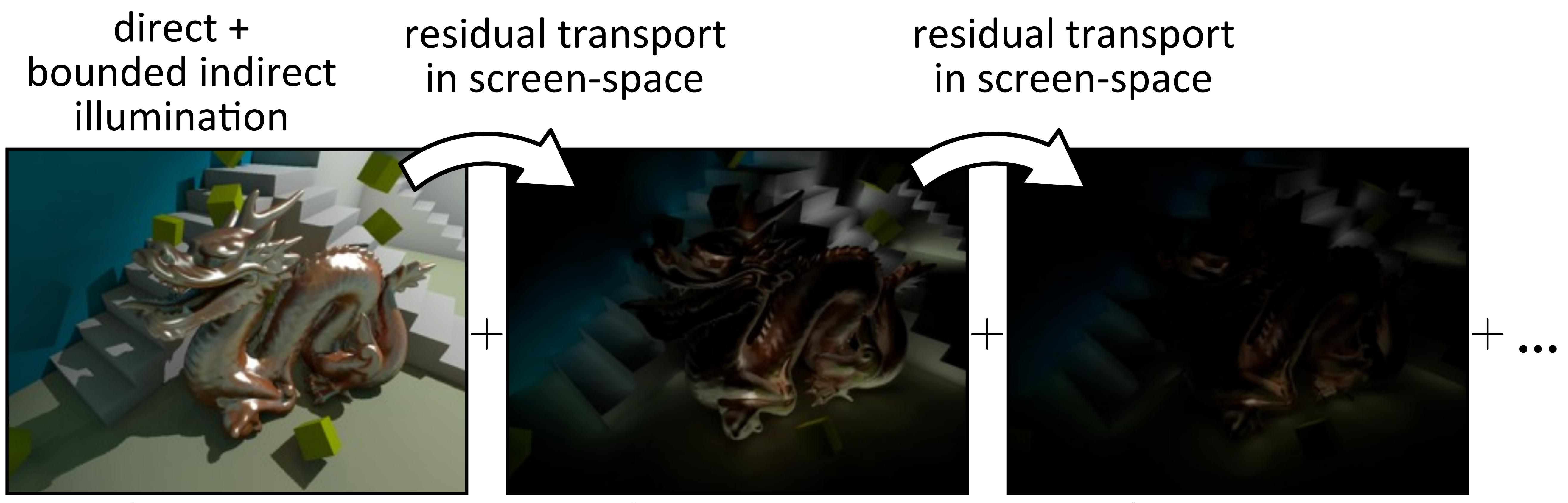
Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process

- Rendering Equation:

$$\approx L_e + \sum_{i=0}^{\infty} \mathbf{T}_{\mathbf{r}}^{i}(\mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L})$$


direct + bounded indirect illumination
residual transport in screen-space
residual transport in screen-space
+
+
+ ...

$\mathbf{T}_{\mathbf{r}}^{0}(\mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L})$ $\mathbf{T}_{\mathbf{r}}^{1}(\mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L})$ $\mathbf{T}_{\mathbf{r}}^{2}(\mathbf{T}L_e + \mathbf{T}_{\mathbf{b}}\hat{L})$

Here is one specific example:

We first render the direct and bounded indirect illumination.

And then we use this image as the input for the computing the residual transport in screen-space. The result of this is then added to the bounded solution and also used as the input for another screen-space integration step, which yields an extra bounce of the residual transport.

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process

We found that 2 steps are usually sufficient, and since the technique is effectively an image filter, it can be implemented on the GPU and further accelerated using hierarchical integration.

The main advantage of this approach is that it is very fast.

The bad side of things is that since we use only the information available in the raster, it is approximate and needs to be conservative otherwise artifacts may occur.

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process

Implementation:

- 2 steps are usually sufficient
- GPU-friendly, hierarchical screen-space integration

We found that 2 steps are usually sufficient, and since the technique is effectively an image filter, it can be implemented on the GPU and further accelerated using hierarchical integration.

The main advantage of this approach is that it is very fast.

The bad side of things is that since we use only the information available in the raster, it is approximate and needs to be conservative otherwise artifacts may occur.

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process

Implementation:

- 2 steps are usually sufficient
- GPU-friendly, hierarchical screen-space integration

Advantages:

- fast! (1024x768 @ 20-30 milliseconds)
- can be implemented as an image filter

We found that 2 steps are usually sufficient, and since the technique is effectively an image filter, it can be implemented on the GPU and further accelerated using hierarchical integration.

The main advantage of this approach is that it is very fast.

The bad side of things is that since we use only the information available in the raster, it is approximate and needs to be conservative otherwise artifacts may occur.

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]

- residual transport is localized and can be applied in post-process

Implementation:

- 2 steps are usually sufficient
- GPU-friendly, hierarchical screen-space integration

Advantages:

- fast! (1024x768 @ 20-30 milliseconds)
- can be implemented as an image filter

Disadvantages:

- approximative, uses information from surfaces visible to camera only
- must be conservative, otherwise artifacts can occur

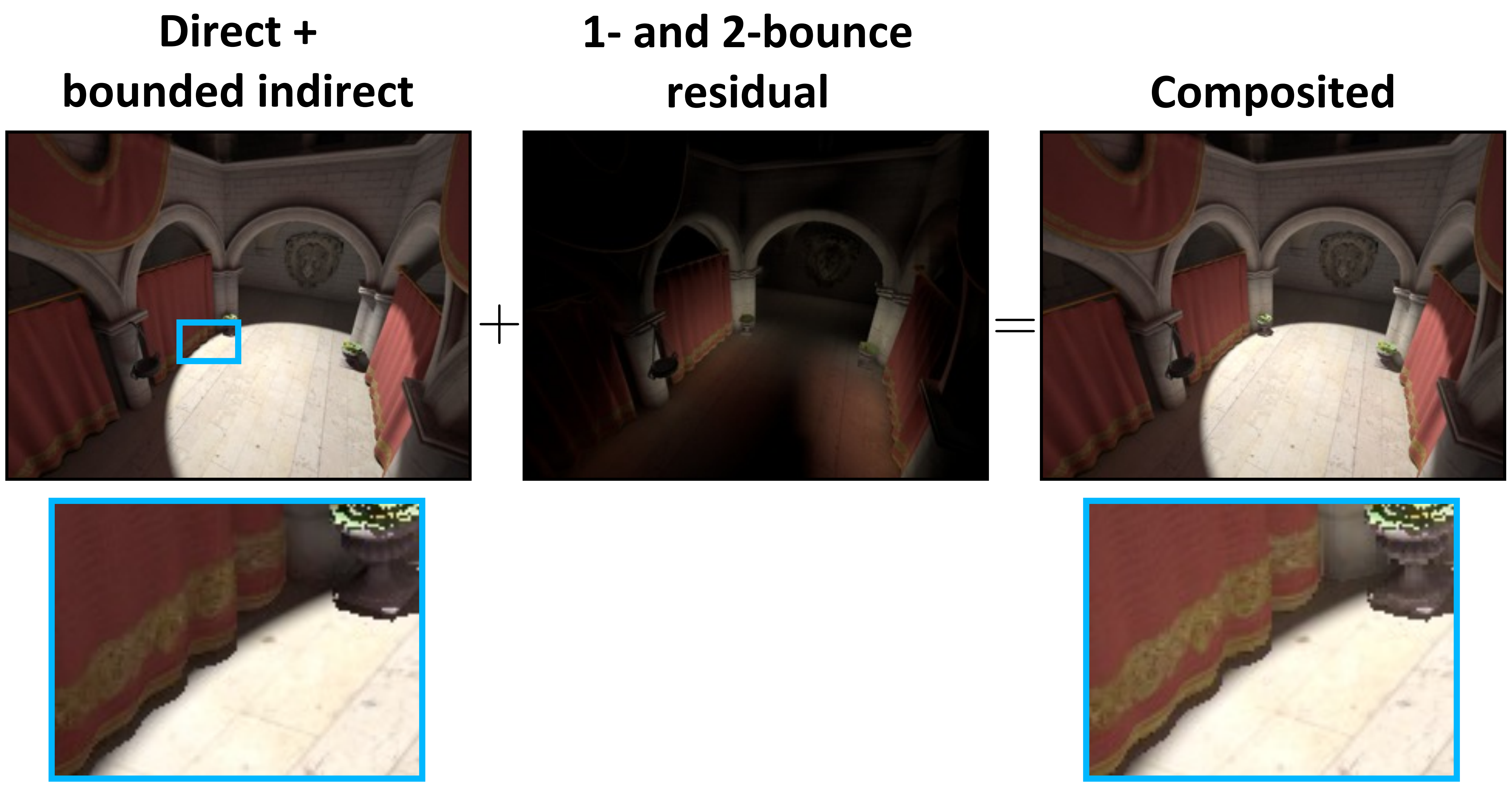
We found that 2 steps are usually sufficient, and since the technique is effectively an image filter, it can be implemented on the GPU and further accelerated using hierarchical integration.

The main advantage of this approach is that it is very fast.

The bad side of things is that since we use only the information available in the raster, it is approximate and needs to be conservative otherwise artifacts may occur.

Bounding & Compensation

Screen-space Bias Compensation [Novák et al. 2011]



Let's quickly look at some results...

you can see that the residual energy recovered using 2 steps of the screen-space bias makes a noticeable difference in the final composite.

Bounding & Compensation

We also looked into the problem of bias compensation for participating media.

This paper was published last year and it contains a lot of analysis and comparisons, which are however fairly specific and thus I will only show some measurements that generalize to most of the VPL algorithms.

Here we see a reference rendering of a rising smoke.

And these two are VPL rendering with unbounded and bounded geometry term. You can see that the artifacts in the media are more severe than on surfaces and the bounding removes really a lot of energy.

Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

We also looked into the problem of bias compensation for participating media.

This paper was published last year and it contains a lot of analysis and comparisons, which are however fairly specific and thus I will only show some measurements that generalize to most of the VPL algorithms.

Here we see a reference rendering of a rising smoke.

And these two are VPL rendering with unbounded and bounded geometry term. You can see that the artifacts in the media are more severe than on surfaces and the bounding removes really a lot of energy.

Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

Reference

We also looked into the problem of bias compensation for participating media.

This paper was published last year and it contains a lot of analysis and comparisons, which are however fairly specific and thus I will only show some measurements that generalize to most of the VPL algorithms.

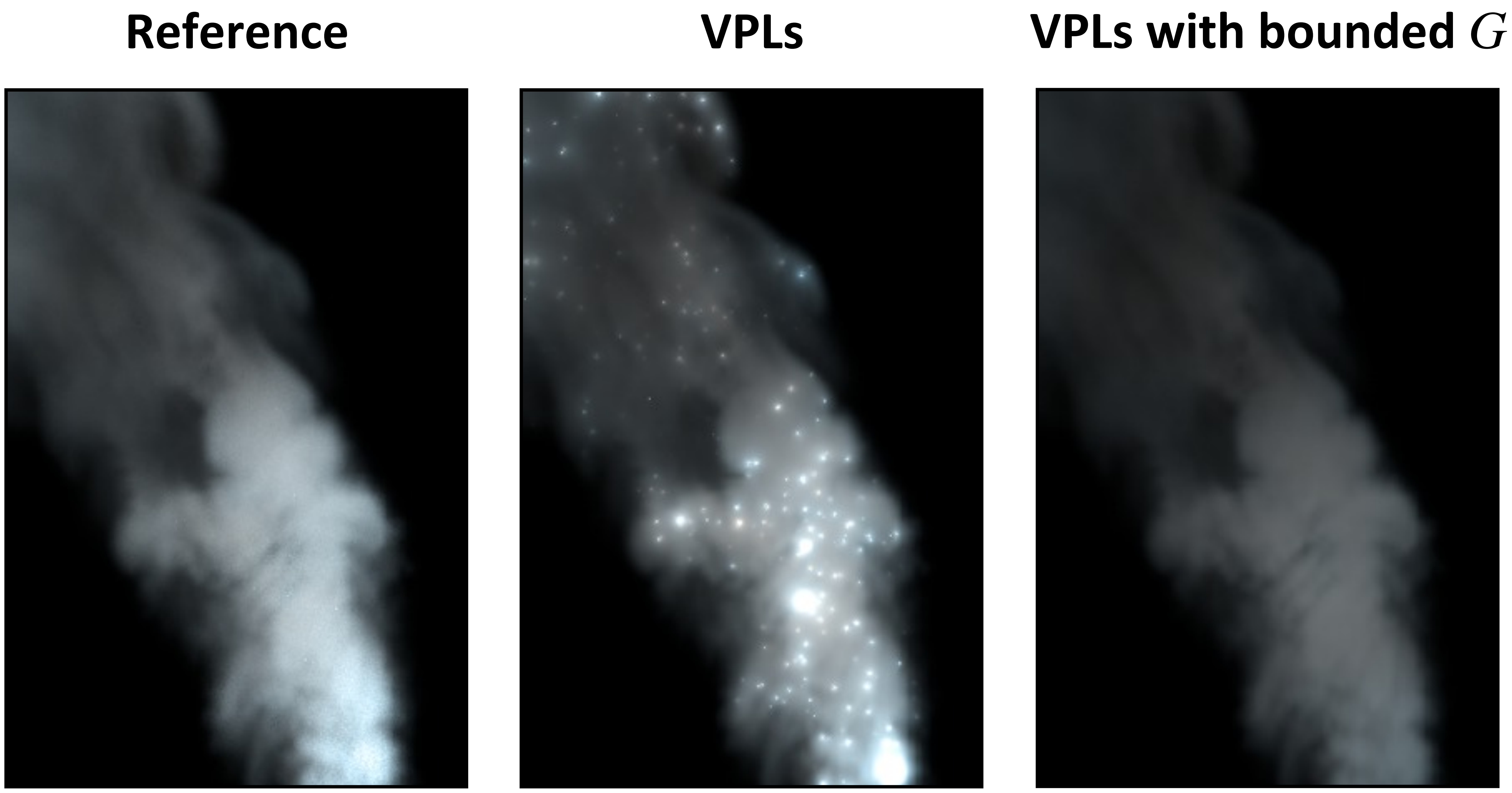
Here we see a reference rendering of a rising smoke.

And these two are VPL rendering with unbounded and bounded geometry term. You can see that the artifacts in the media are more severe than on surfaces and the bounding removes really a lot of energy.

Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media



We also looked into the problem of bias compensation for participating media.

This paper was published last year and it contains a lot of analysis and comparisons, which are however fairly specific and thus I will only show some measurements that generalize to most of the VPL algorithms.

Here we see a reference rendering of a rising smoke.

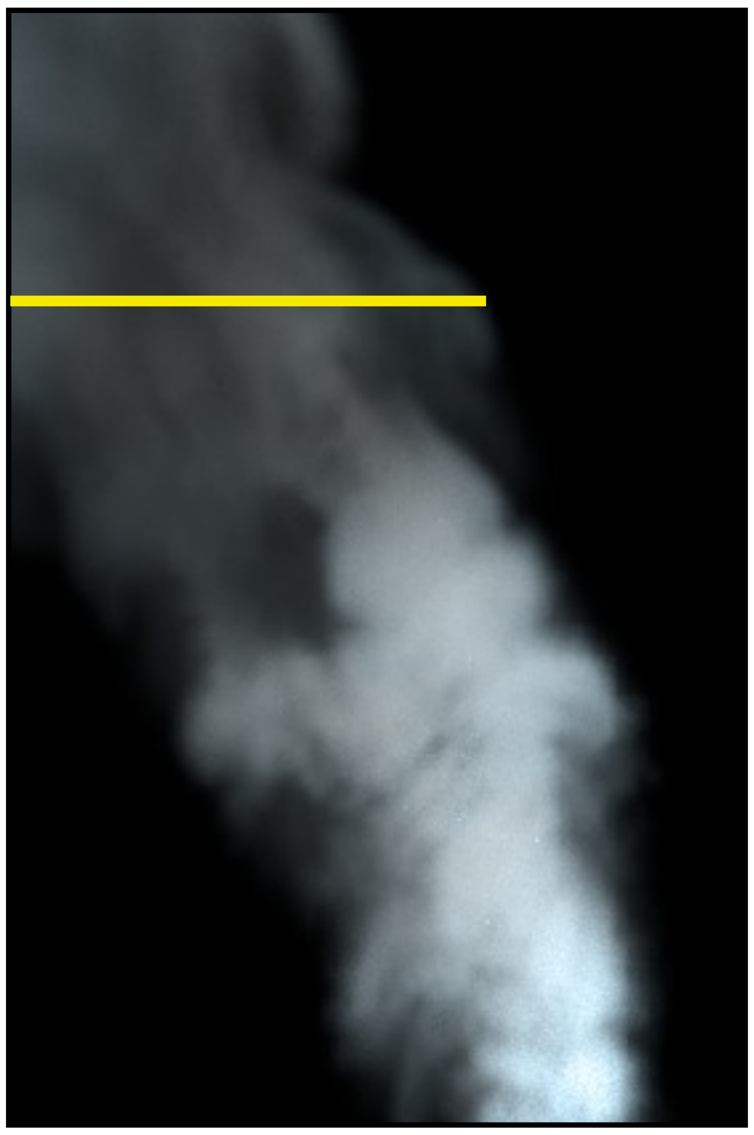
And these two are VPL rendering with unbounded and bounded geometry term. You can see that the artifacts in the media are more severe than on surfaces and the bounding removes really a lot of energy.

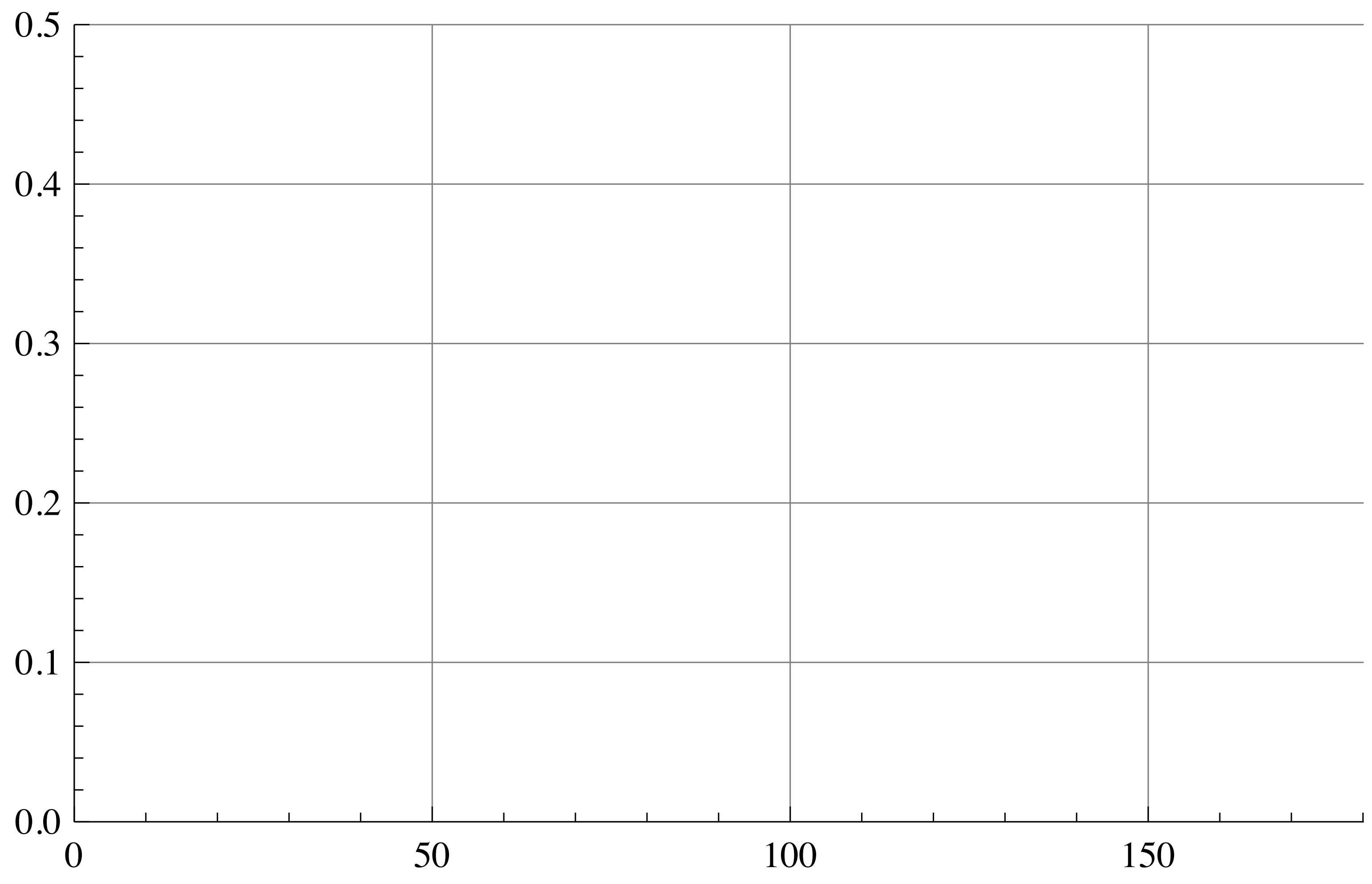
Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

Reference





So let's inspect it more in detail. I will now plot the pixel intensity along the yellow line.

Here we have the reference in green.

This yellow curve is for unbounded VPLs, you can see the spikes, they correspond to the high intensity splotches in the image.

Bounding the geometry term removes the spikes but also looses a lot of energy.

We can add this back by compensating for the bounding, this red curve corresponds to one bounce of the residual transport.

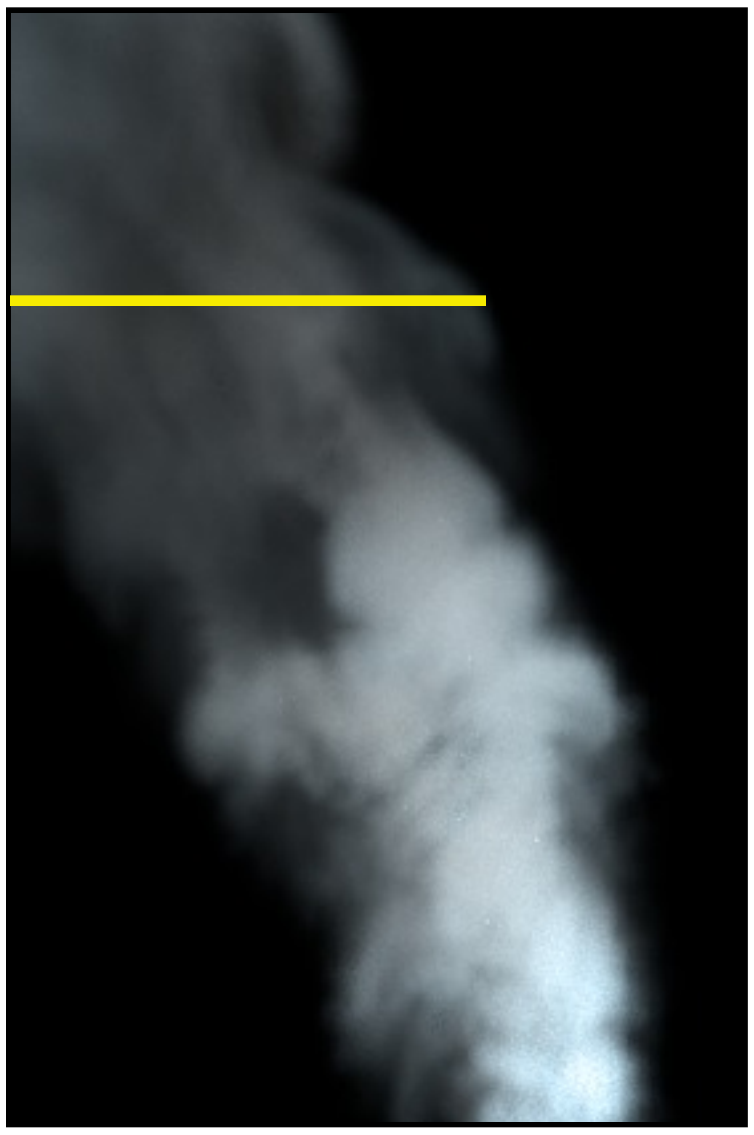
And if we add one more bounce, we get pretty close to the reference already. (This was the motivation in the screen-space bias compensation paper for using only the steps.)

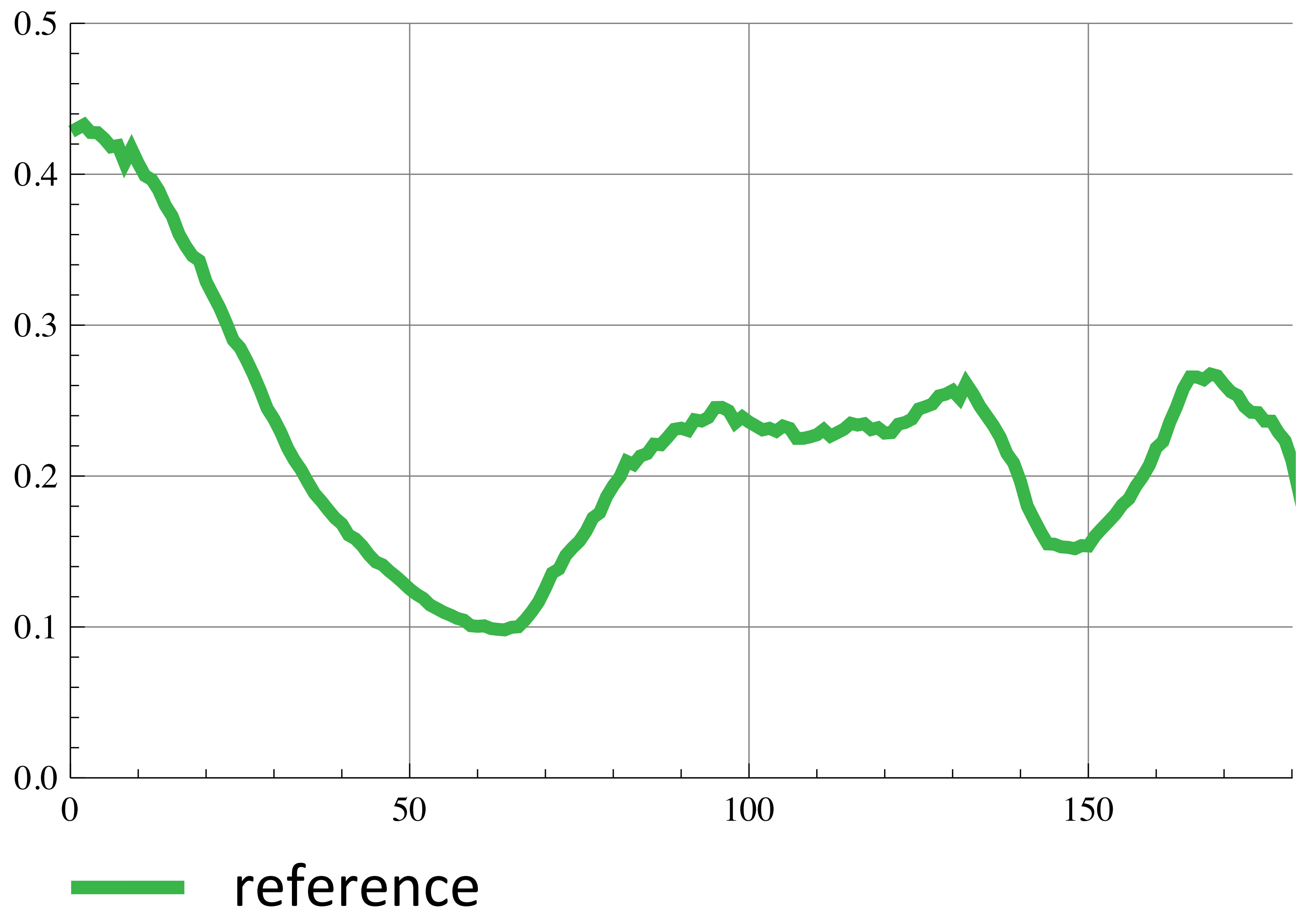
Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

Reference





So let’s inspect it more in detail. I will now plot the pixel intensity along the yellow line.

Here we have the reference in green.

This yellow curve is for unbounded VPLs, you can see the spikes, they correspond to the high intensity splotches in the image.

Bounding the geometry term removes the spikes but also looses a lot of energy.

We can add this back by compensating for the bounding, this red curve corresponds to one bounce of the residual transport.

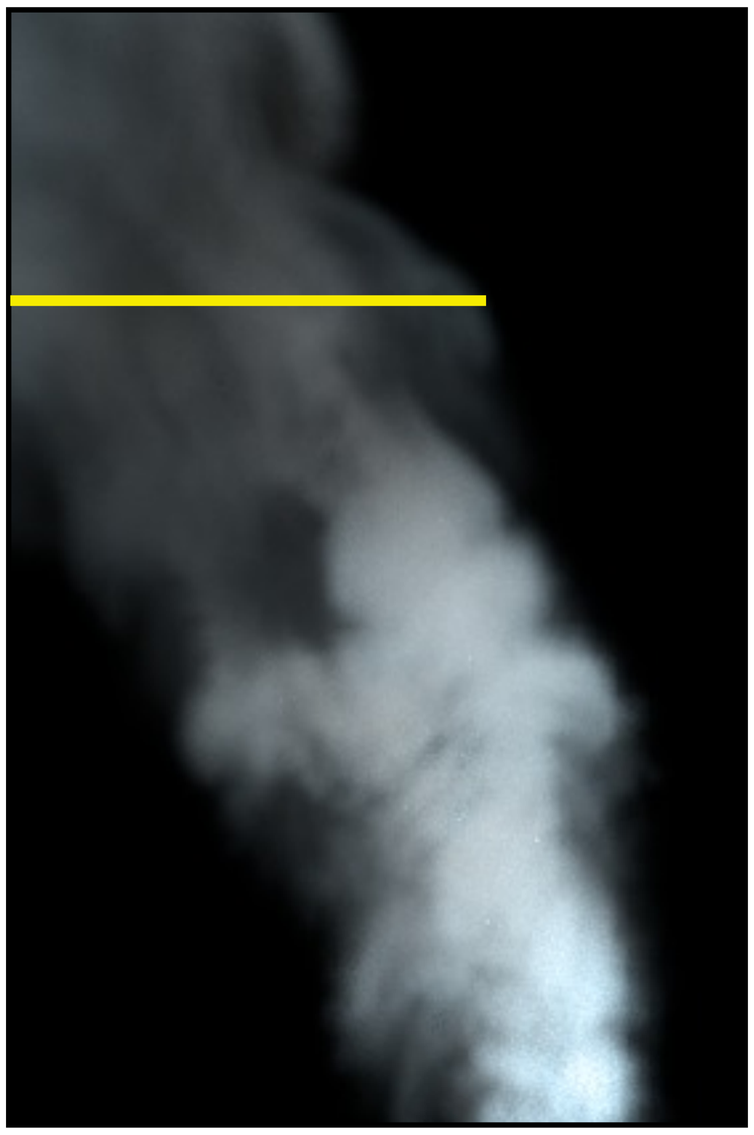
And if we add one more bounce, we get pretty close to the reference already. (This was the motivation in the screen-space bias compensation paper for using only the steps.)

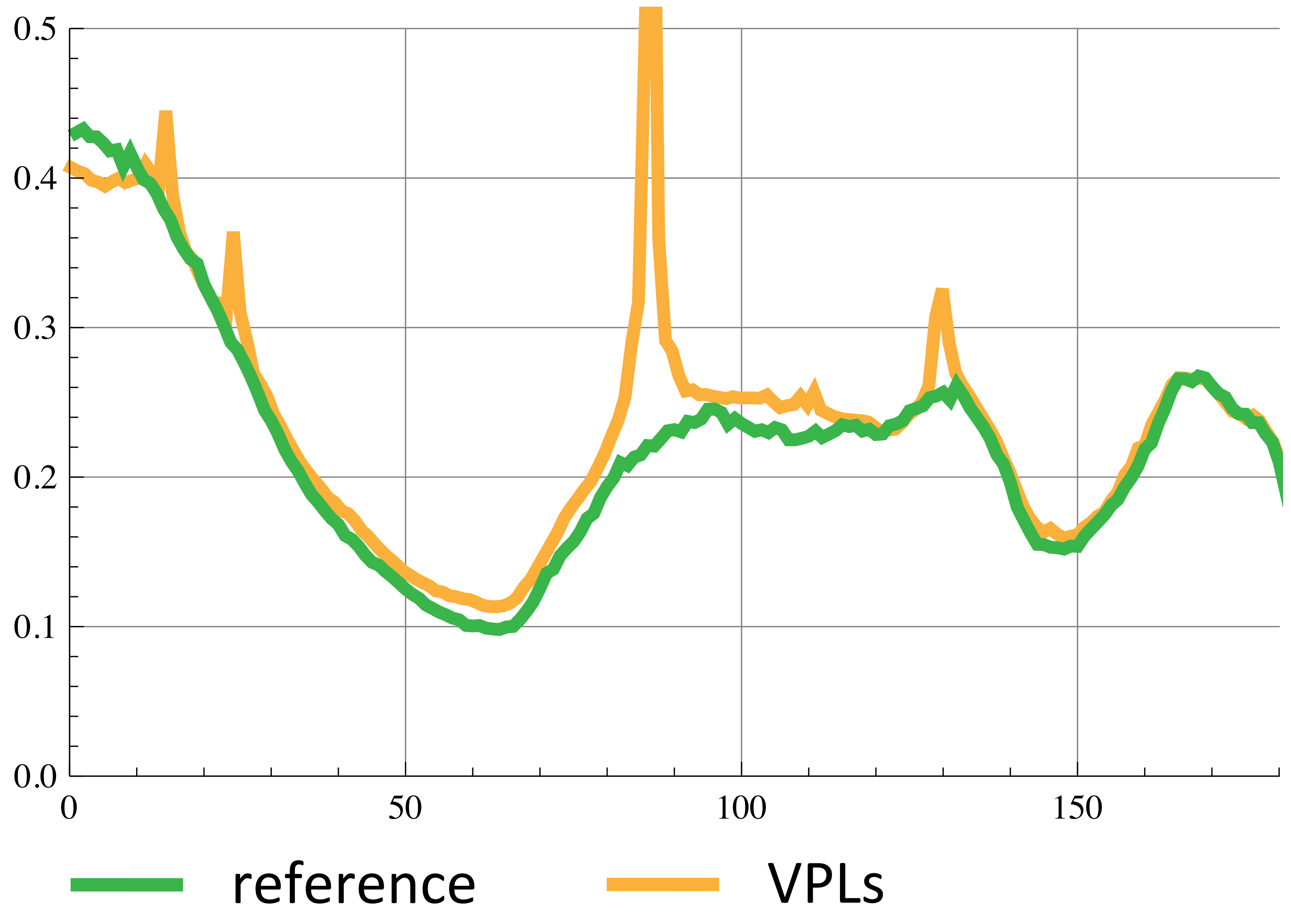
Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

Reference





So let’s inspect it more in detail. I will now plot the pixel intensity along the yellow line.

Here we have the reference in green.

This yellow curve is for unbounded VPLs, you can see the spikes, they correspond to the high intensity splotches in the image.

Bounding the geometry term removes the spikes but also looses a lot of energy.

We can add this back by compensating for the bounding, this red curve corresponds to one bounce of the residual transport.

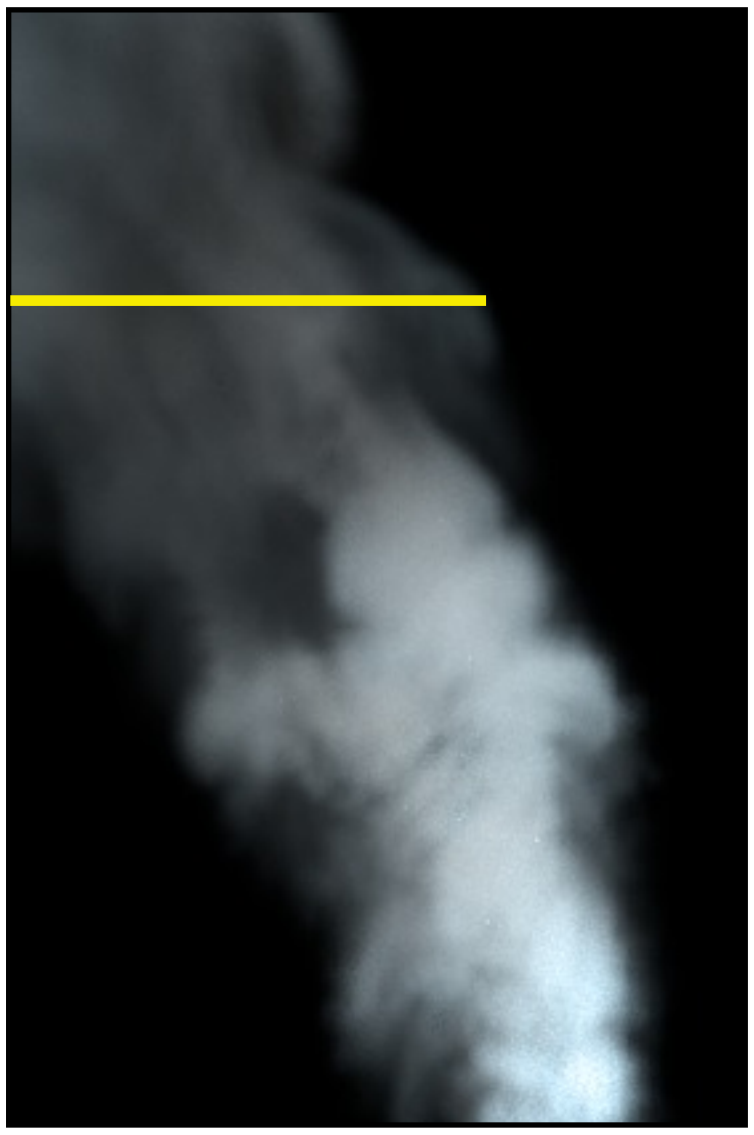
And if we add one more bounce, we get pretty close to the reference already. (This was the motivation in the screen-space bias compensation paper for using only the steps.)

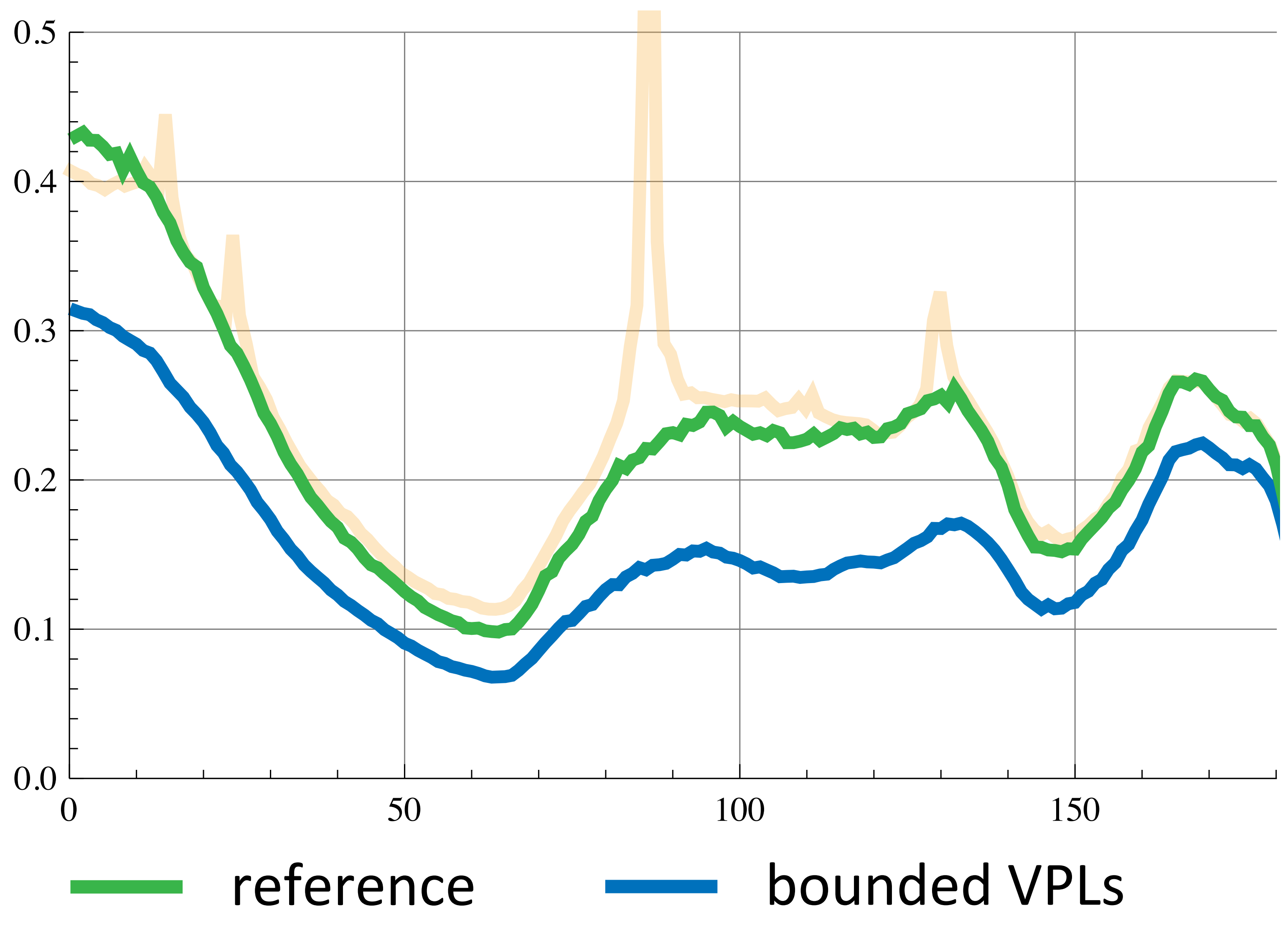
Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

Reference





So let’s inspect it more in detail. I will now plot the pixel intensity along the yellow line.

Here we have the reference in green.

This yellow curve is for unbounded VPLs, you can see the spikes, they correspond to the high intensity splotches in the image.

Bounding the geometry term removes the spikes but also looses a lot of energy.

We can add this back by compensating for the bounding, this red curve corresponds to one bounce of the residual transport.

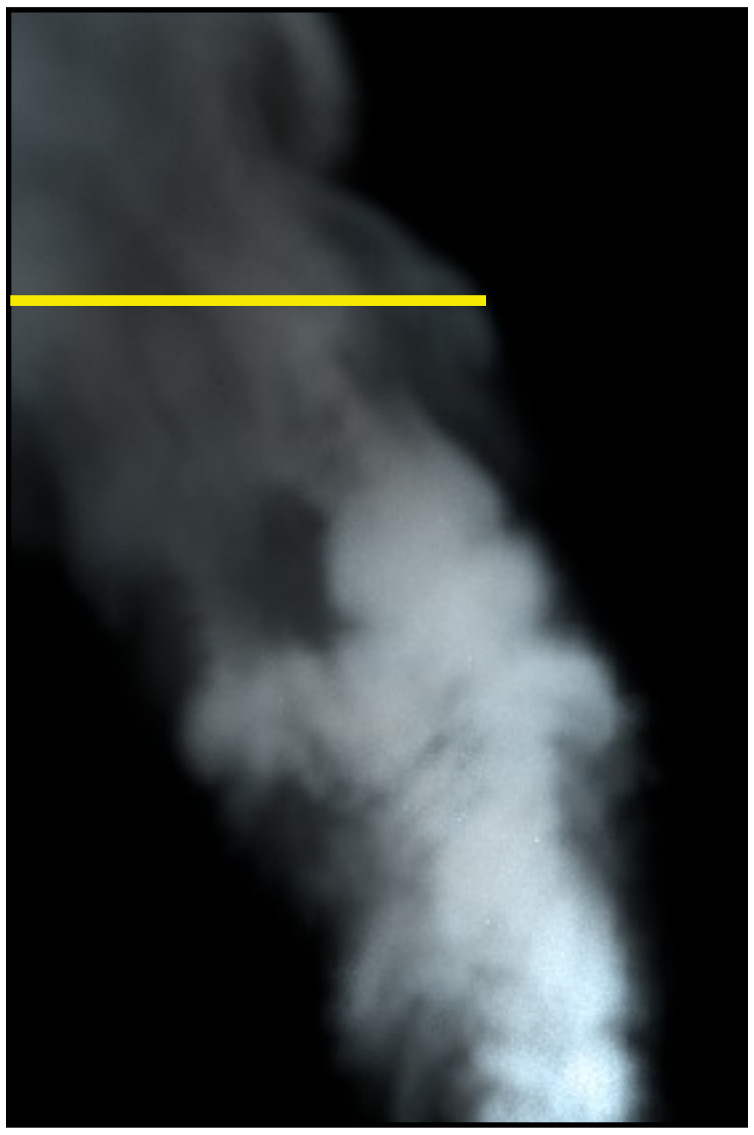
And if we add one more bounce, we get pretty close to the reference already. (This was the motivation in the screen-space bias compensation paper for using only the steps.)

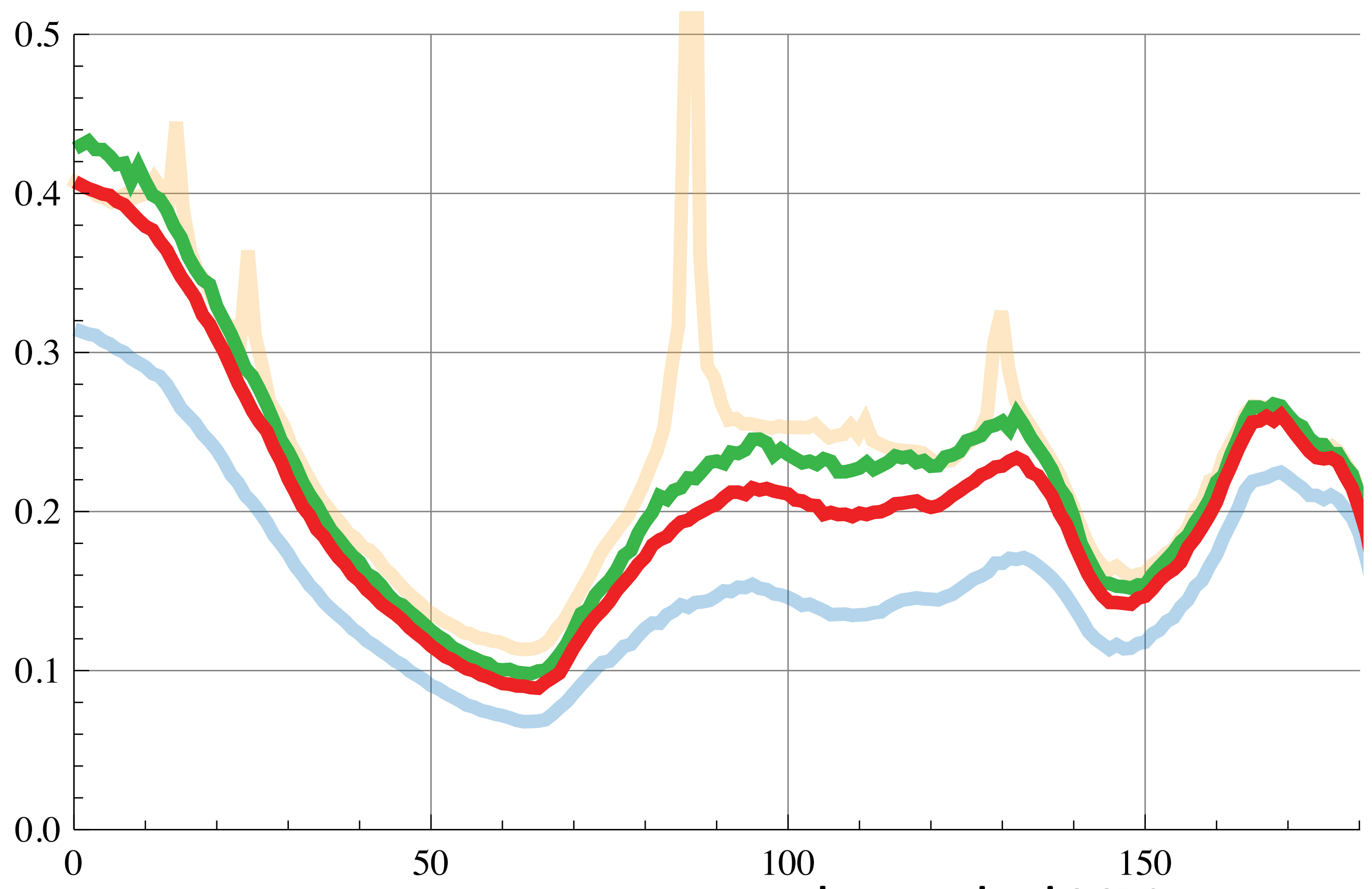
Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

Reference





🟩 reference 🟥 bounded VPLs + 1 bounce of BC

So let’s inspect it more in detail. I will now plot the pixel intensity along the yellow line.

Here we have the reference in green.

This yellow curve is for unbounded VPLs, you can see the spikes, they correspond to the high intensity splotches in the image.

Bounding the geometry term removes the spikes but also looses a lot of energy.

We can add this back by compensating for the bounding, this red curve corresponds to one bounce of the residual transport.

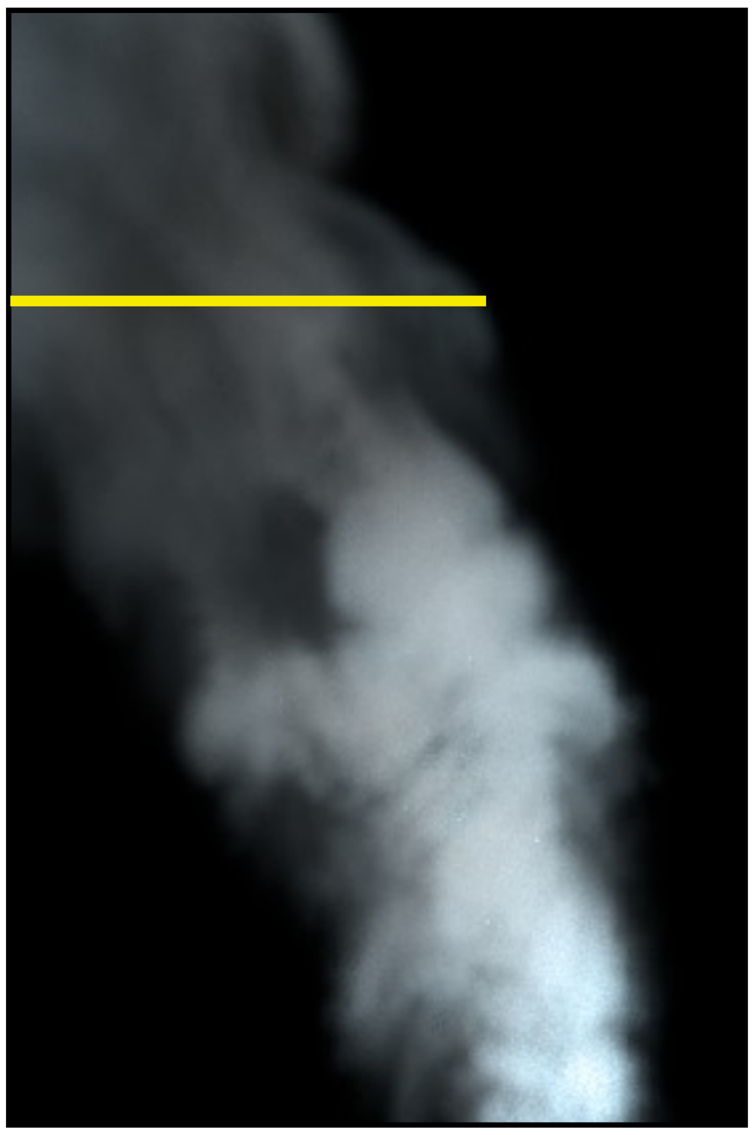
And if we add one more bounce, we get pretty close to the reference already. (This was the motivation in the screen-space bias compensation paper for using only the steps.)

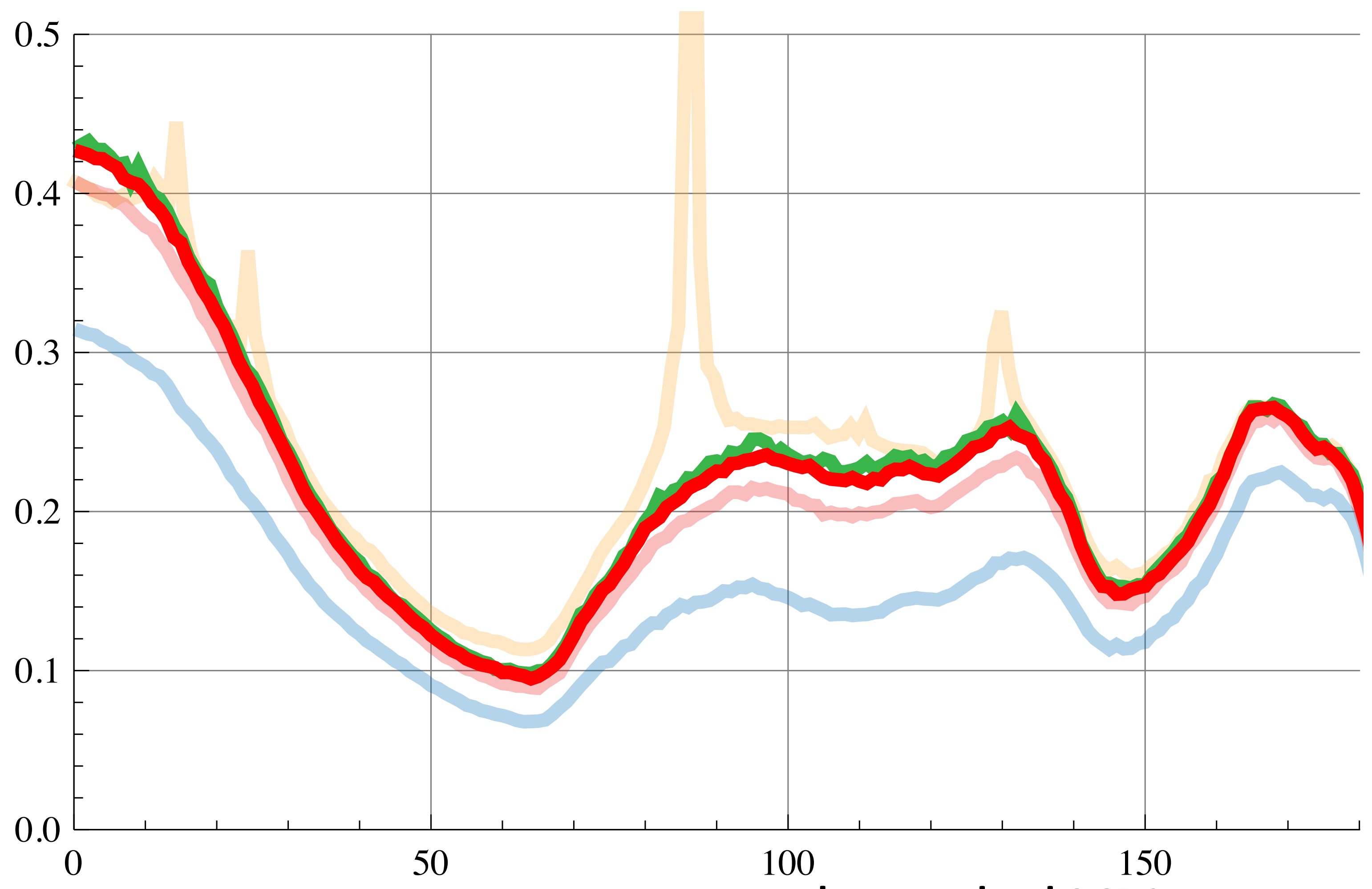
Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

Reference





🟩 reference 🟥 bounded VPLs + 2 bounces of BC

So let's inspect it more in detail. I will now plot the pixel intensity along the yellow line.

Here we have the reference in green.

This yellow curve is for unbounded VPLs, you can see the spikes, they correspond to the high intensity splotches in the image.

Bounding the geometry term removes the spikes but also looses a lot of energy.

We can add this back by compensating for the bounding, this red curve corresponds to one bounce of the residual transport.

And if we add one more bounce, we get pretty close to the reference already. (This was the motivation in the screen-space bias compensation paper for using only the steps.)

Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

As I said, I will not go into the details, they can be found in the paper...

based on the analysis and several other measurements we found that assuming the medium to be locally homogeneous just for the compensation makes the computation of the residual transport much faster and so does omitting the visibility tests.

The technique is relatively fast and GPU friendly, it is however again only approximate and fairly convoluted.

Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

Optimizations used for BC:

- assume locally homogeneous media
- omit testing local visibility

As I said, I will not go into the details, they can be found in the paper...

based on the analysis and several other measurements we found that assuming the medium to be locally homogeneous just for the compensation makes the computation of the residual transport much faster and so does omitting the visibility tests.

The technique is relatively fast and GPU friendly, it is however again only approximate and fairly convoluted.

Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

Optimizations used for BC:

- assume locally homogeneous media
- omit testing local visibility

Advantages:

- fast, GPU friendly

As I said, I will not go into the details, they can be found in the paper...

based on the analysis and several other measurements we found that assuming the medium to be locally homogeneous just for the compensation makes the computation of the residual transport much faster and so does omitting the visibility tests.

The technique is relatively fast and GPU friendly, it is however again only approximate and fairly convoluted.

Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

- efficient compensation for participating media

Optimizations used for BC:

- assume locally homogeneous media
- omit testing local visibility

Advantages:

- fast, GPU friendly

Disadvantages:

- approximate, complicated

As I said, I will not go into the details, they can be found in the paper...

based on the analysis and several other measurements we found that assuming the medium to be locally homogeneous just for the compensation makes the computation of the residual transport much faster and so does omitting the visibility tests.

The technique is relatively fast and GPU friendly, it is however again only approximate and fairly convoluted.

Bounding & Compensation

Approximate Bias Compensation [Engelhardt et al. 2012]

bounded: 39 min.
approx. bias comp.: 13 min.

Here you can see an example rendering, computing the bounded transport took 39 minutes, the approximate bias compensation only 13 minutes.

Lighting with VPLs

Splotches!!!

Solutions:

1. **Bound the geometry term**
   - removes energy, darkens the image
   - to get unbiased results, we need to compensate for the bounding

   [Kollig and Keller 2004], [Raab et al. 2008], [Davidovič et al. 2010], [Novák et al. 2011], [Engelhardt et al. 2012]

2. **Distribute the flux of a VPL over area (volume)**
   - redistributes energy, blurs the illumination
   - to get consistent results, progressively reduce the blurring

   [Hašan et al. 2009], [Novák et al. 2012a], [Novák et al. 2012b]

Ok, so these were four techniques that avoid the splotches by first bounding the geometry term and then trying to recover the removed energy using a different integration scheme.

The other class of techniques tries to avoid the splotches by spatially spreading the energy so that it is no longer concentrated at single points

Lighting with VPLs

Splotches!!!

Solutions:

1. **Bound the geometry term**
   - removes energy, darkens the image
   - to get unbiased results, we need to compensate for the bounding

   [Kollig and Keller 2004], [Raab et al. 2008], [Davidovič et al. 2010], [Novák et al. 2011], [Engelhardt et al. 2012]

2. **Distribute the flux of a VPL over area (volume)**
   - redistributes energy, blurs the illumination
   - to get consistent results, progressively reduce the blurring

   [Hašan et al. 2009], [Novák et al. 2012a], [Novák et al. 2012b]

Ok, so these were four techniques that avoid the splotches by first bounding the geometry term and then trying to recover the removed energy using a different integration scheme.

The other class of techniques tries to avoid the splotches by spatially spreading the energy so that it is no longer concentrated at single points

Spreading the Energy

Virtual Spherical Lights [Hašan et al. 2009]

- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere

The first one to propose this was Hašan and colleagues at Siggraph Asia 2009 who presented a new lighting primitive called virtual spherical light.

Here we have an illustration showing a shading point and a VPL.

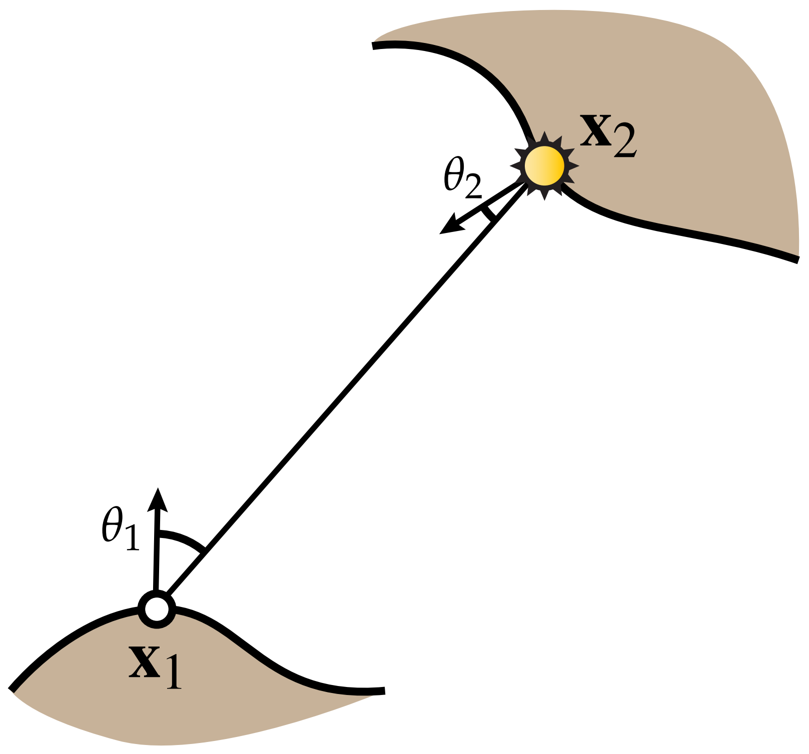
And the light that reaches the shading point from the VPL is defined by this equation.

These are the terms that cause the splotches.

Spreading the Energy

Virtual Spherical Lights [Hašan et al. 2009]

- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere



The first one to propose this was Hašan and colleagues at Siggraph Asia 2009 who presented a new lighting primitive called virtual spherical light.

Here we have an illustration showing a shading point and a VPL.

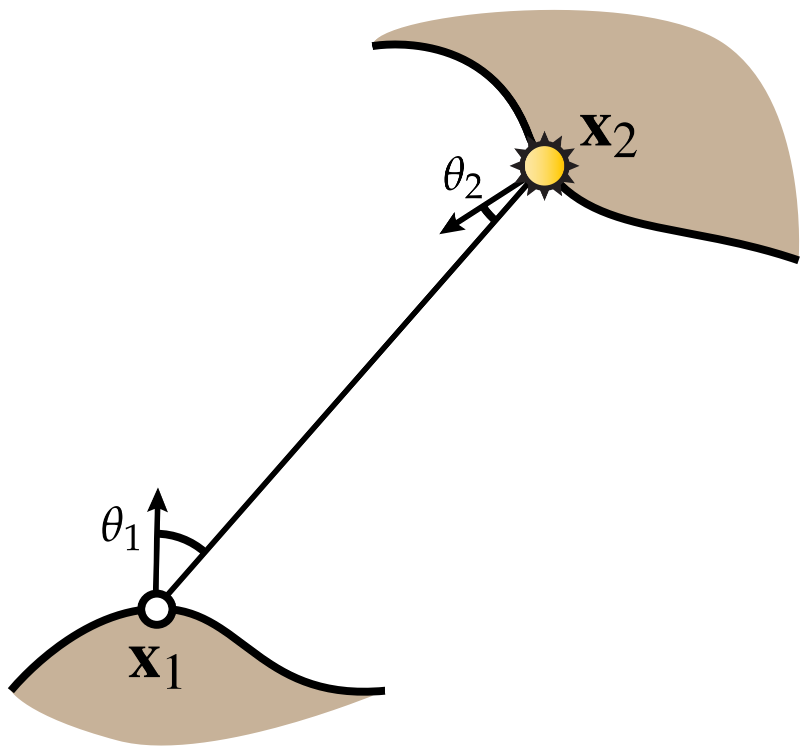
And the light that reaches the shading point from the VPL is defined by this equation.

These are the terms that cause the splotches.

Spreading the Energy

Virtual Spherical Lights [Hašan et al. 2009]

- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere



point-to-point: $$\Phi\, V(\mathbf{x}_1, \mathbf{x}_2)\, f(\mathbf{x}_1)\, f(\mathbf{x}_2)\, \frac{\cos\theta_1 \cos\theta_2}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

The first one to propose this was Hašan and colleagues at Siggraph Asia 2009 who presented a new lighting primitive called virtual spherical light.

Here we have an illustration showing a shading point and a VPL.

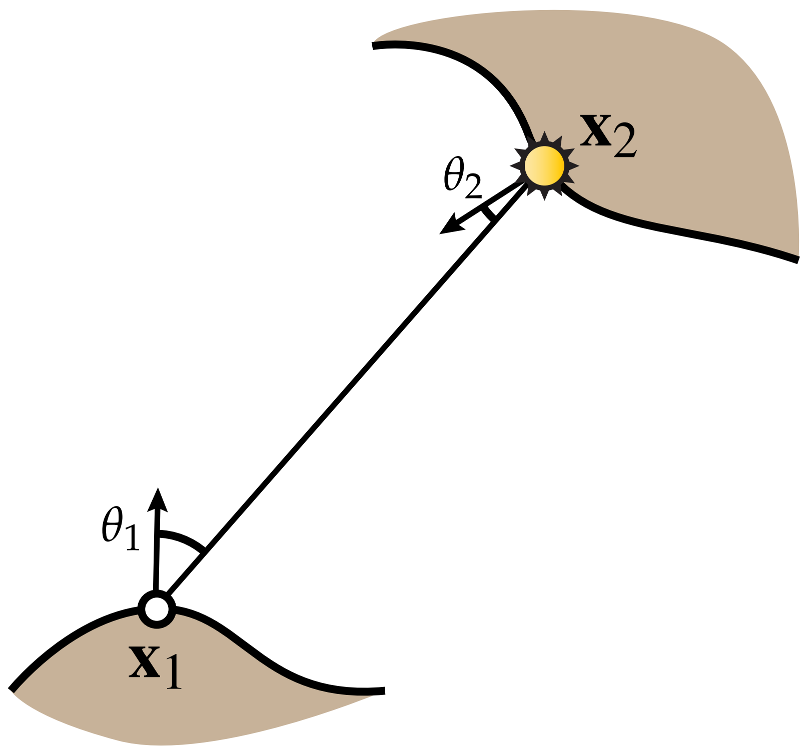
And the light that reaches the shading point from the VPL is defined by this equation.

These are the terms that cause the splotches.

Spreading the Energy

Virtual Spherical Lights [Hašan et al. 2009]

- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere



splotches!

point-to-point: $$\Phi\, V(\mathbf{x}_1,\mathbf{x}_2)\, f(\mathbf{x}_1)\, f(\mathbf{x}_2)\, \frac{\cos\theta_1 \cos\theta_2}{\|\mathbf{x}_1-\mathbf{x}_2\|^2}$$

The first one to propose this was Hašan and colleagues at Siggraph Asia 2009 who presented a new lighting primitive called virtual spherical light.

Here we have an illustration showing a shading point and a VPL.

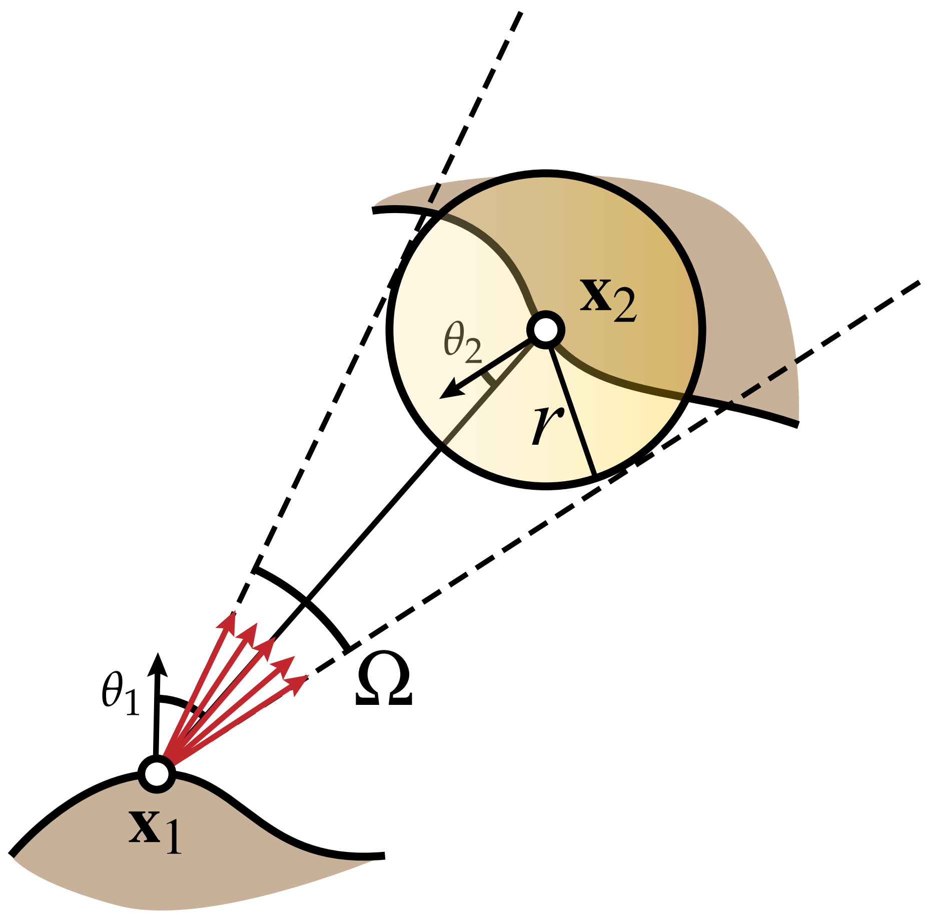
And the light that reaches the shading point from the VPL is defined by this equation.

These are the terms that cause the splotches.

Spreading the Energy

Virtual Spherical Lights [Hašan et al. 2009]

- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere



splotches!

point-to-point: $$\Phi\, V(\mathbf{x}_1, \mathbf{x}_2)\, f(\mathbf{x}_1)\, f(\mathbf{x}_2)\, \frac{\cos\theta_1 \cos\theta_2}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

To reduce the splotches, the authors propose to inflate the point light into a sphere and distribute its flux over the surfaces inside the sphere.

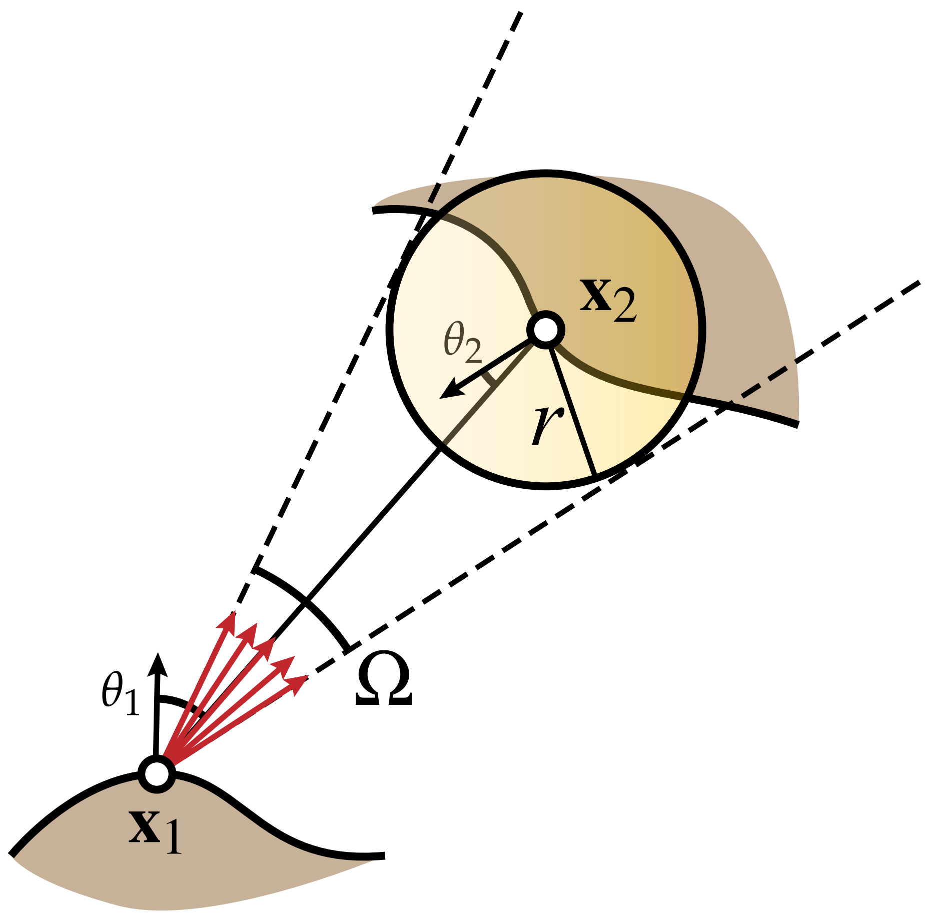
The point-to-point evaluation is thus replaced by an approximation of a sphere-to-point transfer.

You can see that we now integrate over the solid angle Omega subtended by the sphere, which removes the squared distance, and also averages the values of the BRDFs over the solid angle.

Spreading the Energy

Virtual Spherical Lights [Hašan et al. 2009]

- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere



splotches!

point-to-point: $\Phi\, V(\mathbf{x}_1,\mathbf{x}_2)\; f(\mathbf{x}_1)\; f(\mathbf{x}_2)\; \dfrac{\cos\theta_1 \cos\theta_2}{\|\mathbf{x}_1-\mathbf{x}_2\|^2}$

approx. sphere-to-point: $\dfrac{\Phi}{\pi r^2}\, V(\mathbf{x}_1,\mathbf{x}_2) \int_\Omega f(\mathbf{x}_1)\; f(\mathbf{x}_2) \cos\theta_1 \cos\theta_2 \, d\omega$

To reduce the splotches, the authors propose to inflate the point light into a sphere and distribute its flux over the surfaces inside the sphere.

The point-to-point evaluation is thus replaced by an approximation of a sphere-to-point transfer.

You can see that we now integrate over the solid angle Omega subtended by the sphere, which removes the squared distance, and also averages the values of the BRDFs over the solid angle.

Spreading the Energy

Virtual Spherical Lights [Hašan et al. 2009]

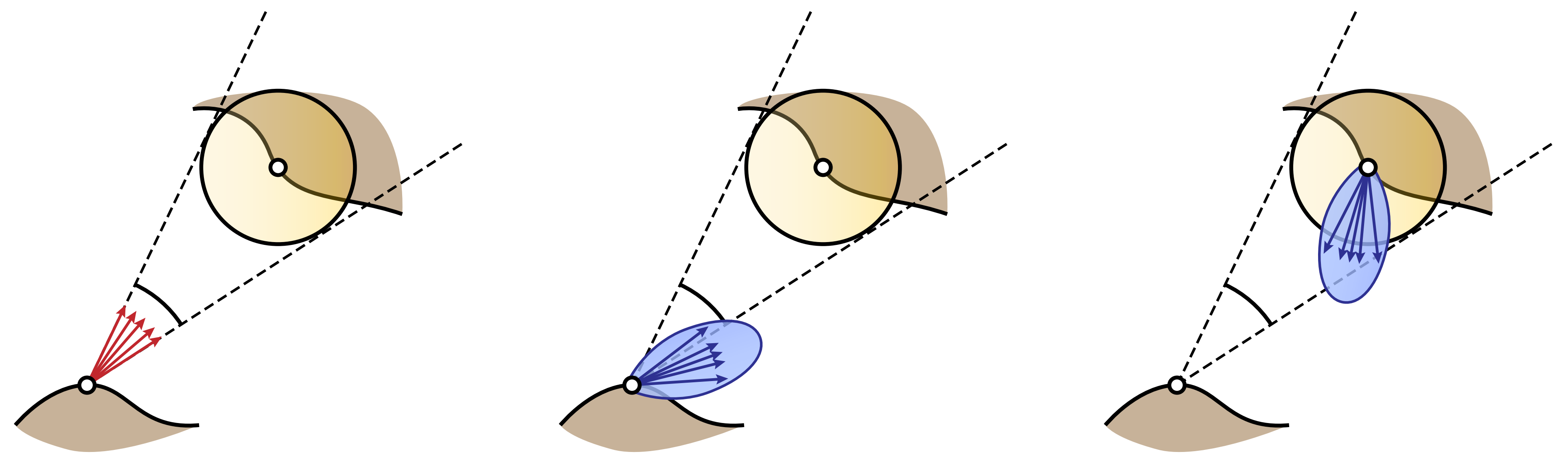
- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere

To reduce the noise due to the numerical integration, they propose to combine sampling of the cone with importance sampling of each of the BRDFs using multiple importance sampling.

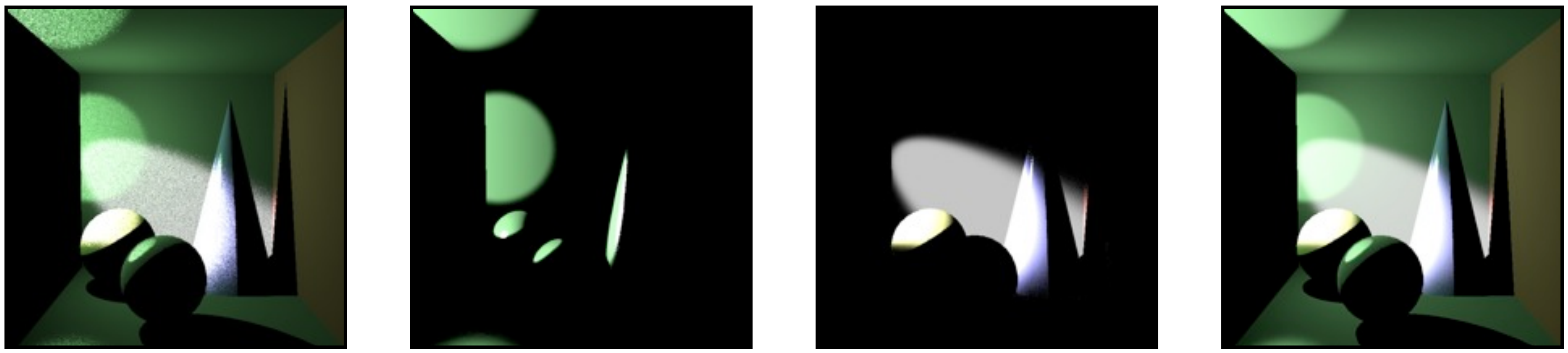
Spreading the Energy

Virtual Spherical Lights [Hašan et al. 2009]

- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere



cone sampling   BRDF1 sampling   BRDF2 sampling   Multiple importance sampling



To reduce the noise due to the numerical integration, they propose to combine sampling of the cone with importance sampling of each of the BRDFs using multiple importance sampling.

Bounding & Compensation

Virtual Spherical Lights [Hašan et al. 2009]

- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere

So the main advantage of spherical lights is that no energy is removed, it is only blurred. This introduces some systematic error, but in general seems to be better than clamping.

The integration also increases the computation time, but can be GPU accelerated.

Bounding & Compensation

Virtual Spherical Lights [Hašan et al. 2009]

- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere

Advantages:

- energy is blurred, not clamped

So the main advantage of spherical lights is that no energy is removed, it is only blurred. This introduces some systematic error, but in general seems to be better than clamping.

The integration also increases the computation time, but can be GPU accelerated.

Bounding & Compensation

Virtual Spherical Lights [Hašan et al. 2009]

- distribute the energy of the infinitesimal VPL over nearby surfaces inside a sphere

Advantages:

- energy is blurred, not clamped

Disadvantages:

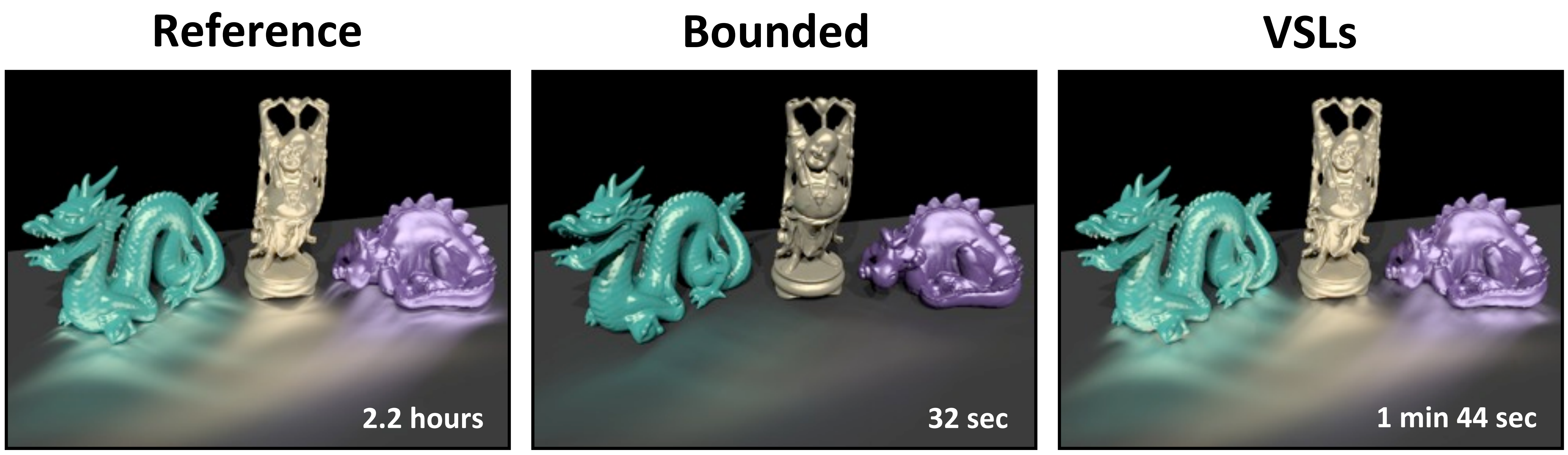
- introduces bias
- requires an extra integration over the solid angle

So the main advantage of spherical lights is that no energy is removed, it is only blurred. This introduces some systematic error, but in general seems to be better than clamping.

The integration also increases the computation time, but can be GPU accelerated.

Spreading the Energy

Virtual Spherical Lights [Hašan et al. 2009]



Here we have a comparison in a scene with highly glossy anisotropic material.

You can see that VSLs preserve the energy, the illumination features are just a bit blurry.

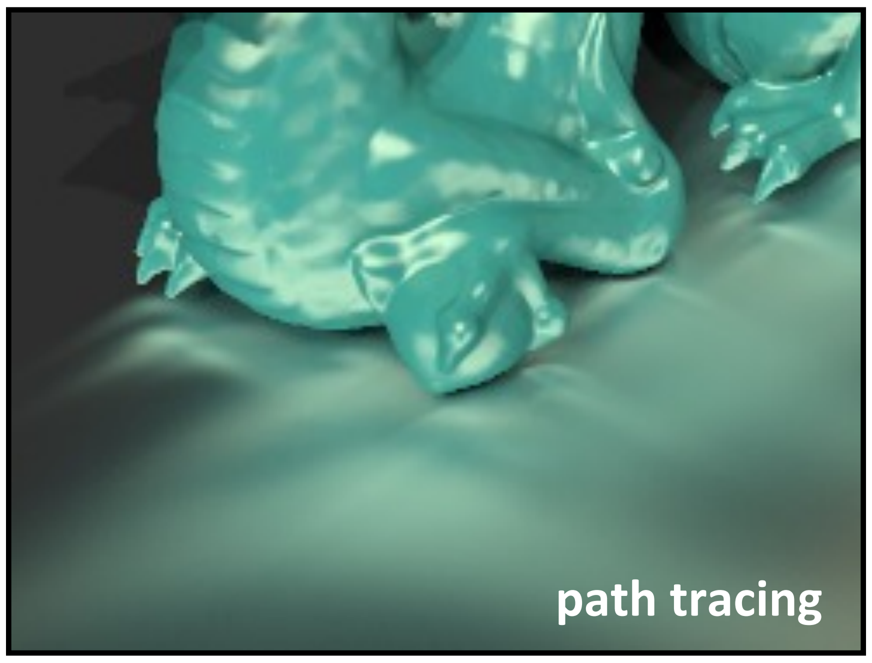
This can be seen in this close-up in the middle image, which uses only 5000 VSLs.

As we increase the number of VSLs, the results get closer to the reference.

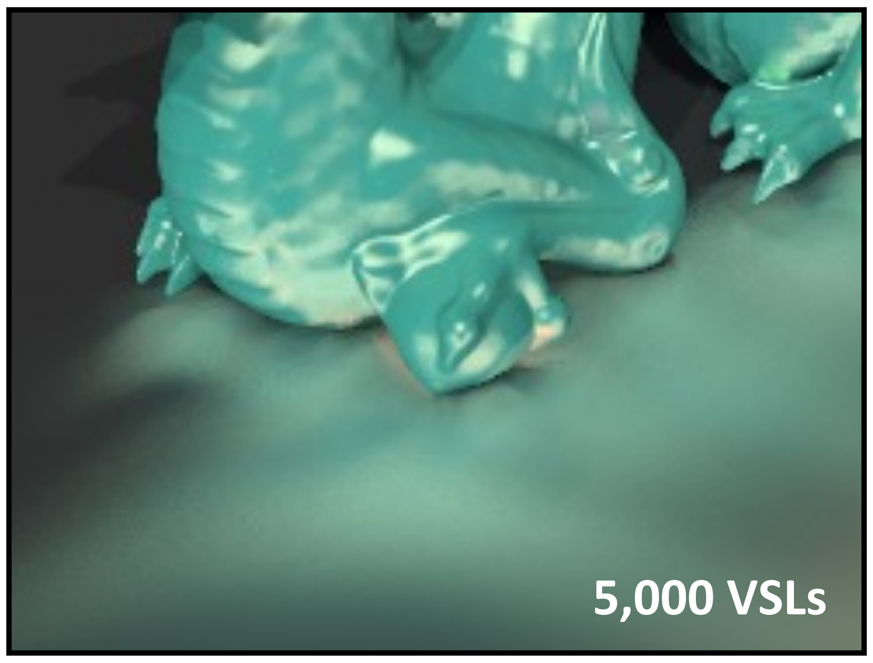
Spreading the Energy

Virtual Spherical Lights [Hašan et al. 2009]

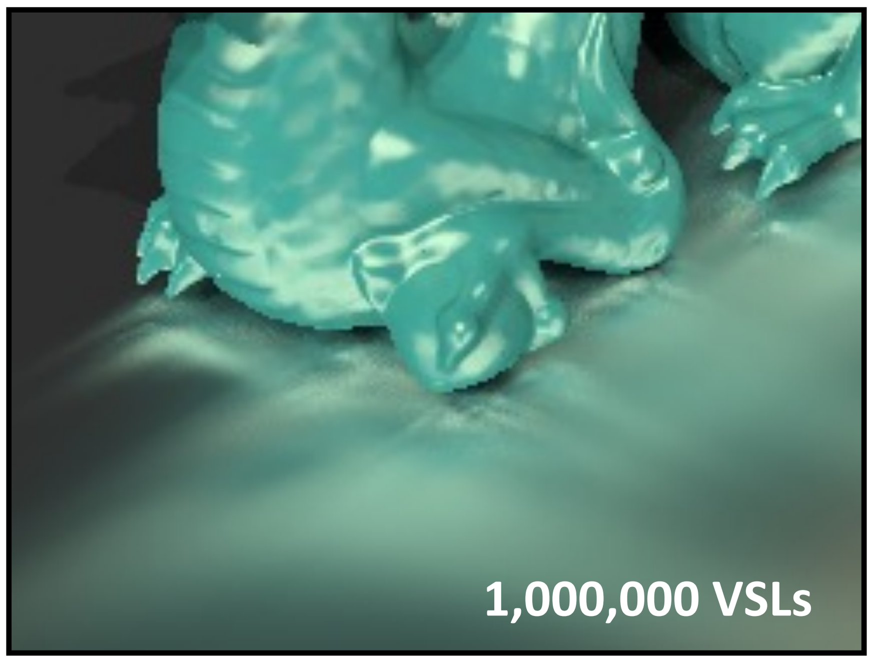
Reference



VSLs



VSLs converged



Here we have a comparison in a scene with highly glossy anisotropic material.

You can see that VSLs preserve the energy, the illumination features are just a bit blurry.

This can be seen in this close-up in the middle image, which uses only 5000 VSLs.

As we increase the number of VSLs, the results get closer to the reference.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources

A great opportunity for spreading the energy arises in participating media, where scattering of light can be formulated continuously along rays.

We leveraged this observation and about a year ago proposed a new lighting primitive called virtual ray lights.

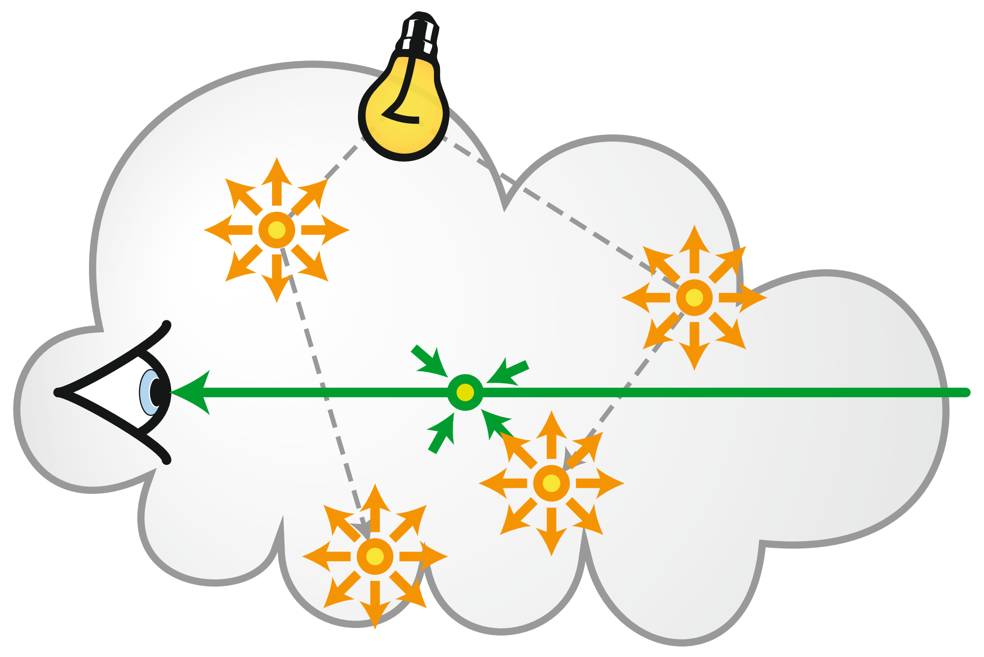
On your left-hand side you can see a few VPLs that were created from vertices of two random walks.
The idea of ray lights is to use entire segments of the walk and turn them into linear light sources.
This yields denser sampling of the path space and provably reduces the degree of the singularity.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources

Virtual Point Lights



A great opportunity for spreading the energy arises in participating media, where scattering of light can be formulated continuously along rays.

We leveraged this observation and about a year ago proposed a new lighting primitive called virtual ray lights.

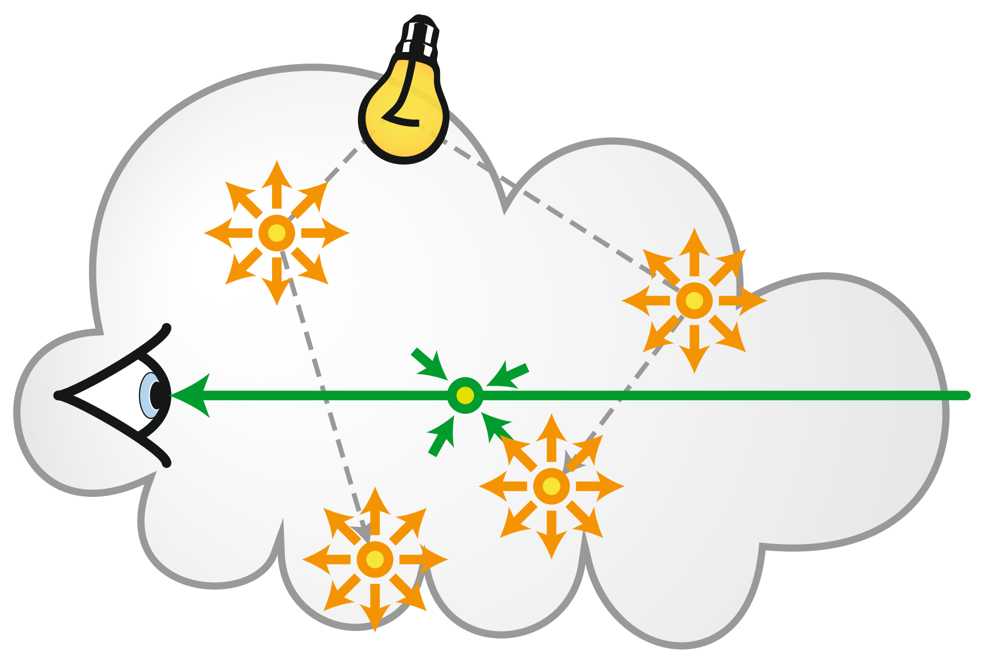
On your left-hand side you can see a few VPLs that were created from vertices of two random walks.
The idea of ray lights is to use entire segments of the walk and turn them into linear light sources.
This yields denser sampling of the path space and provably reduces the degree of the singularity.

Spreading the Energy

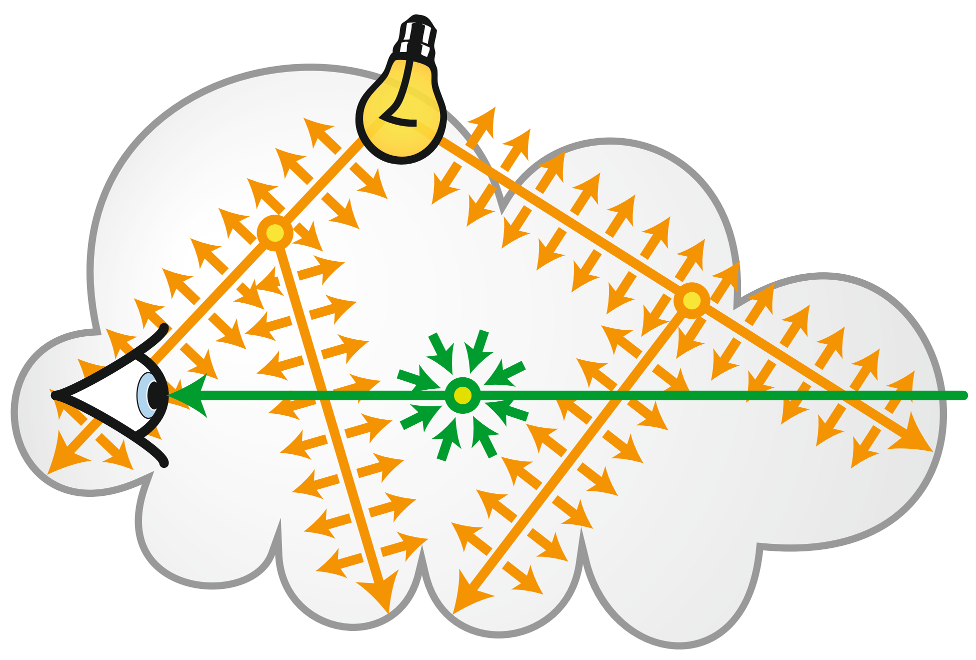
Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources

Virtual Point Lights



Virtual Ray Lights



- higher sampling of path space
- provably reduce singularities

A great opportunity for spreading the energy arises in participating media, where scattering of light can be formulated continuously along rays.

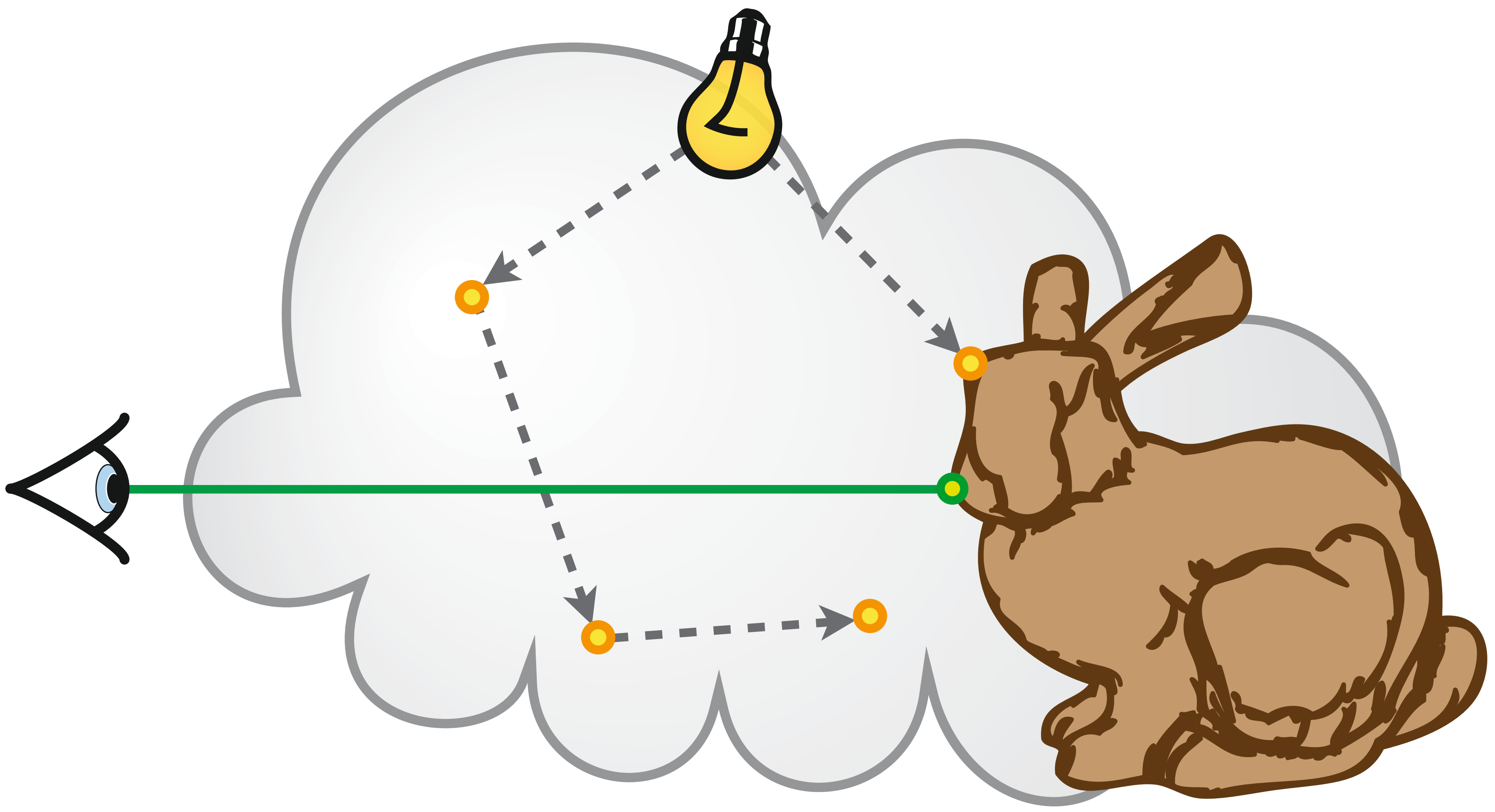
We leveraged this observation and about a year ago proposed a new lighting primitive called virtual ray lights.

On your left-hand side you can see a few VPLs that were created from vertices of two random walks.
The idea of ray lights is to use entire segments of the walk and turn them into linear light sources.
This yields denser sampling of the path space and provably reduces the degree of the singularity.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources



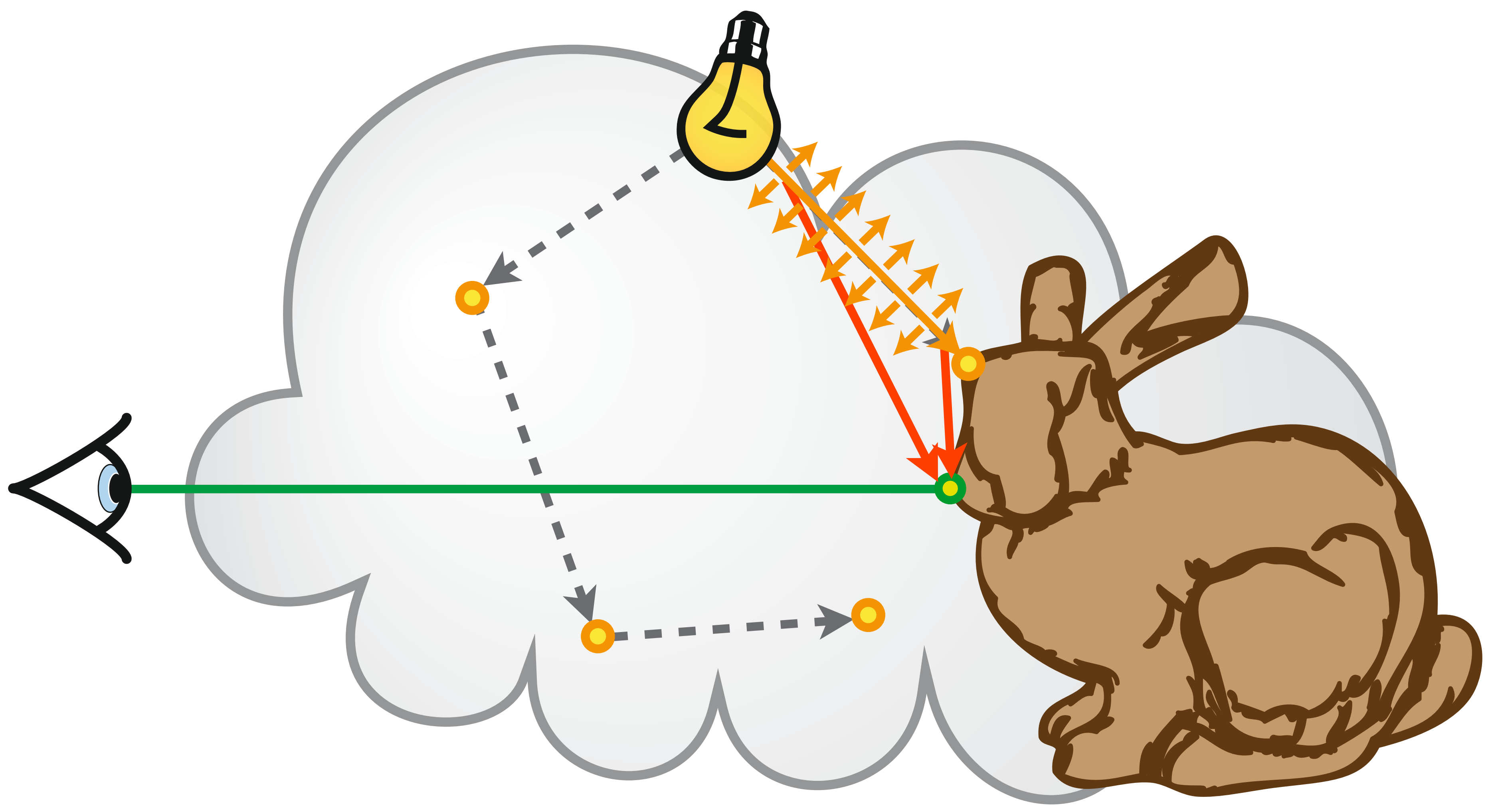
Here is an animation how the algorithm works.

We take each segment, turn it into a light source and compute the contribution to the camera ray and to the surface point.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources



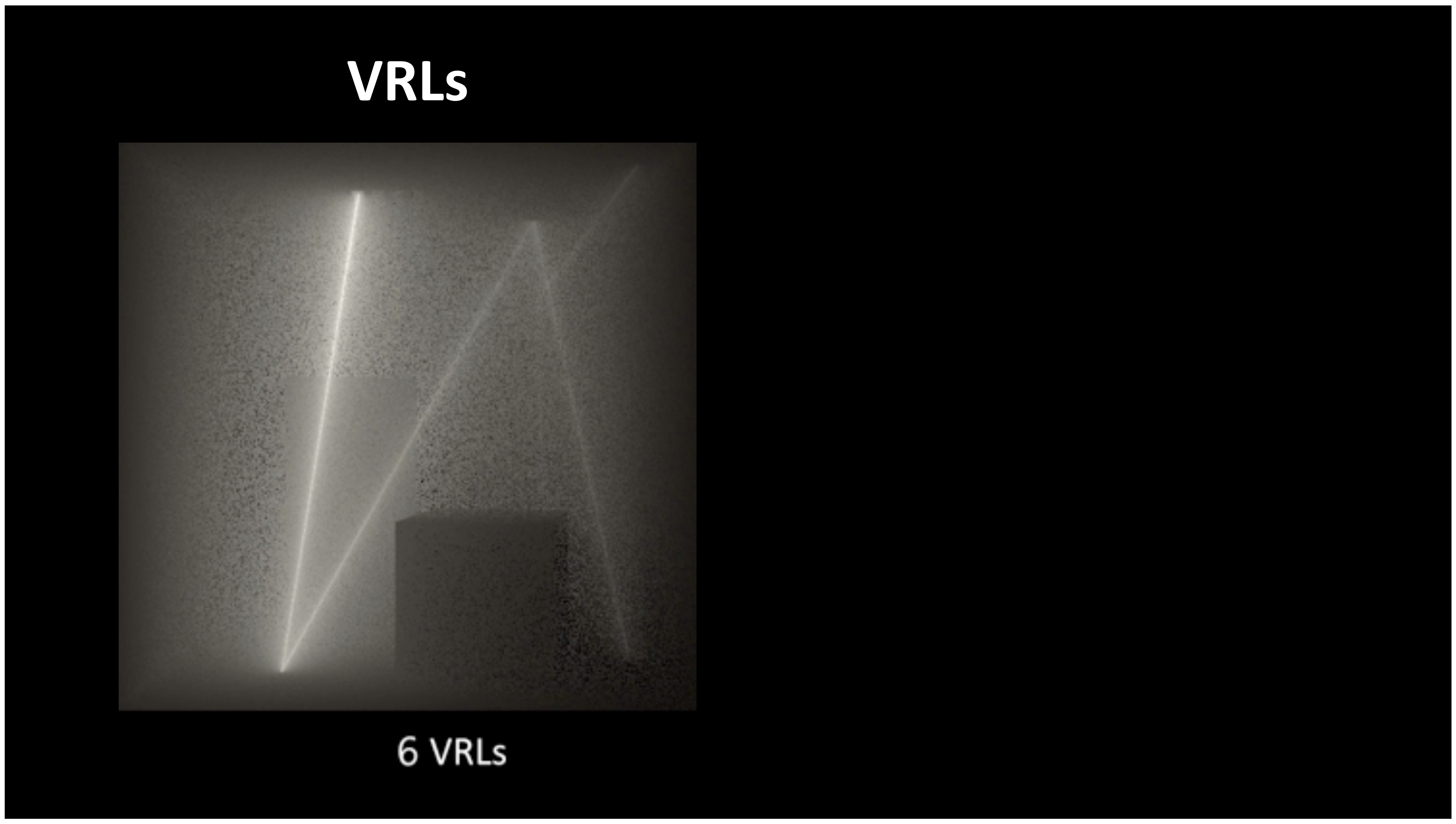
Here is an animation how the algorithm works.

We take each segment, turn it into a light source and compute the contribution to the camera ray and to the surface point.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources



Here we have a progressive rendering of multiple scattering.

You can see the individual ray lights, but they average out quite quickly and there is no bounding applied.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources

VRLs **VPLs**

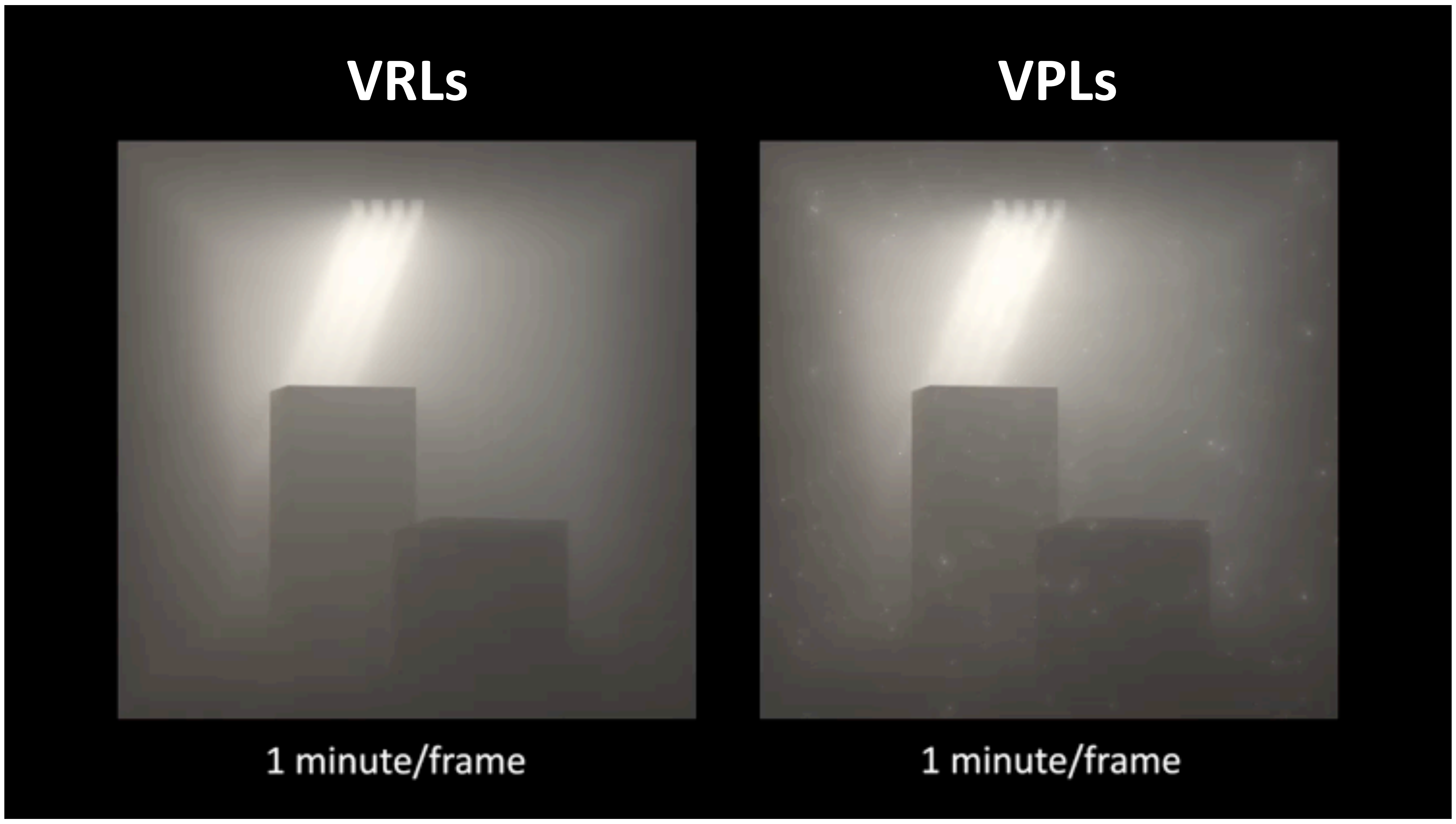
This video compares the temporal stability, which is usually pretty bad with VPLs as you can see in the right sequence.

In contrast, the ray lights on the left do not suffer from any flickering.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources



This video compares the temporal stability, which is usually pretty bad with VPLs as you can see in the right sequence.

In contrast, the ray lights on the left do not suffer from any flickering.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources

The big advantage of ray lights is that the spreading of energy does not introduce any bias, since we take advantage of the continuous scattering of light in media.

The singularity is reduced, although not removed completely.

The major drawback is the slightly more involved integration, but in the paper we provide two suitable importance sampling strategies make it efficient.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources

Advantages:

- energy is spread along lines, singularity is reduced (not removed)
- unbiased, temporally stable

The big advantage of ray lights is that the spreading of energy does not introduce any bias, since we take advantage of the continuous scattering of light in media.

The singularity is reduced, although not removed completely.

The major drawback is the slightly more involved integration, but in the paper we provide two suitable importance sampling strategies make it efficient.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

- many-light technique for participating media
- use segments of the random walk as light sources

Advantages:

- energy is spread along lines, singularity is reduced (not removed)
- unbiased, temporally stable

Disadvantages:

- requires 2D integration (along both rays)

The big advantage of ray lights is that the spreading of energy does not introduce any bias, since we take advantage of the continuous scattering of light in media.

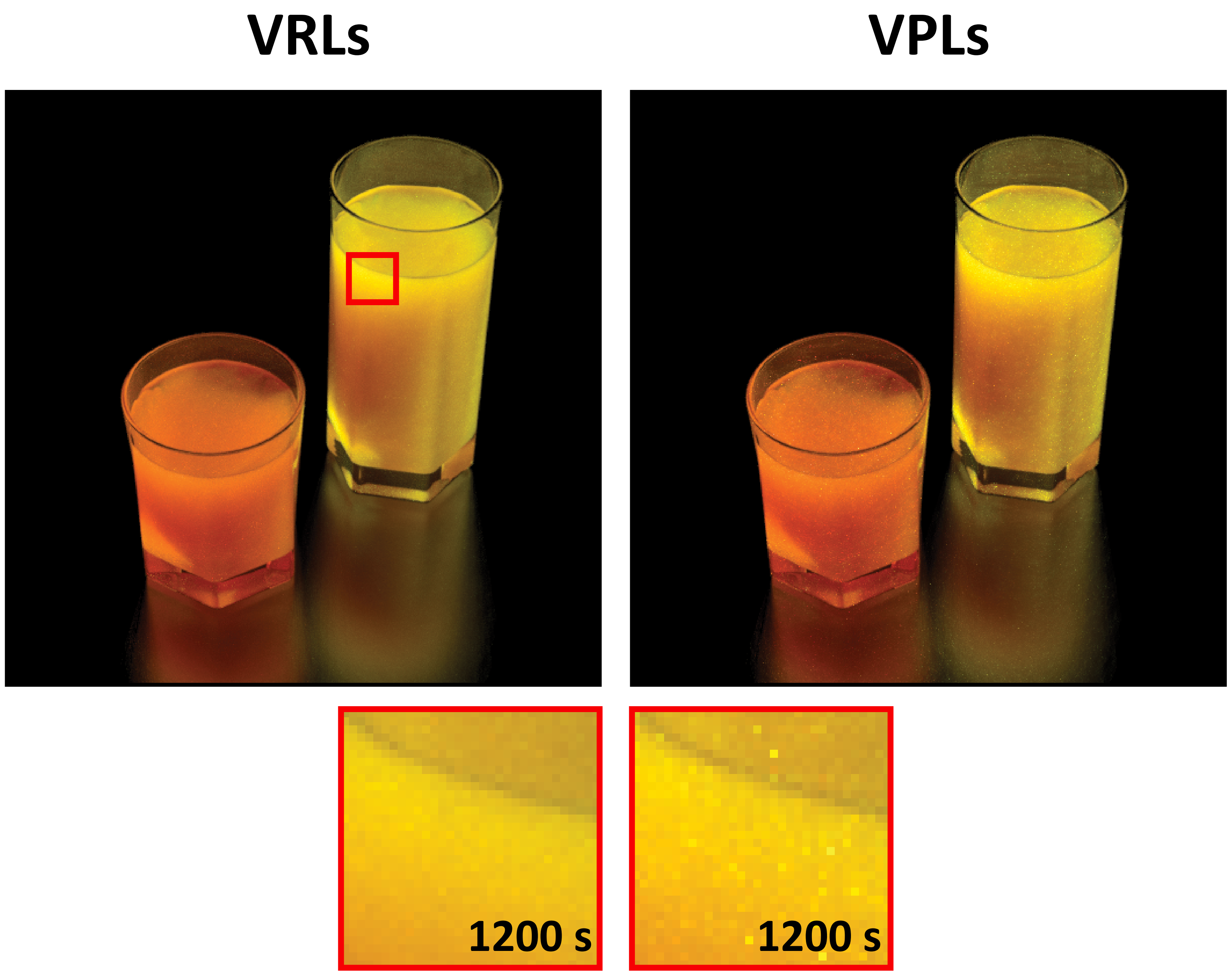
The singularity is reduced, although not removed completely.

The major drawback is the slightly more involved integration, but in the paper we provide two suitable importance sampling strategies make it efficient.

Spreading the Energy

Virtual Ray Lights [Novák et al. 2012a]

Multiple scattering



And here is a comparison of multiple scattering in orange juice.

There was no clamping used in each of the techniques and in the insets you can see that after twenty minutes the ray light rendering is almost perfectly converged, whereas VPLs still suffer from some artifacts.

Spreading the Energy

Progressive Virtual Beam Lights [Novák et al. 2012b]

- give thickness to VRLs
- use the concept of VSLs to “inflate” points on the ray

The last technique that I will mention is a follow up on the ray lights.

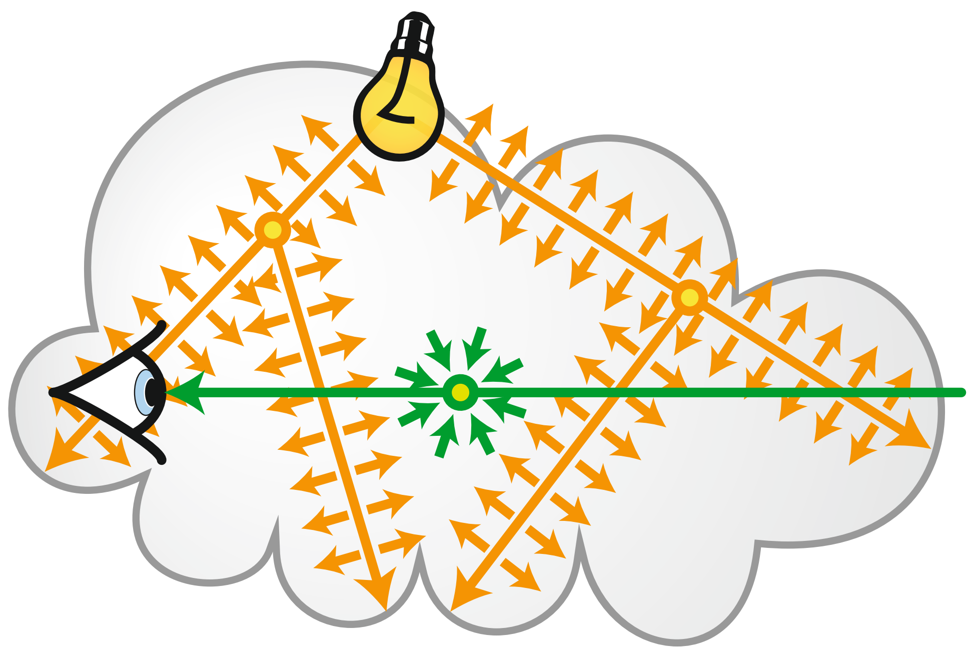
It was inspired by the spherical lights; we basically take each ray light and inflate it into a beam. This removes the singularities completely.

Spreading the Energy

Progressive Virtual Beam Lights [Novák et al. 2012b]

- give thickness to VRLs
- use the concept of VSLs to "inflate" points on the ray

Virtual Ray Lights



The last technique that I will mention is a follow up on the ray lights.

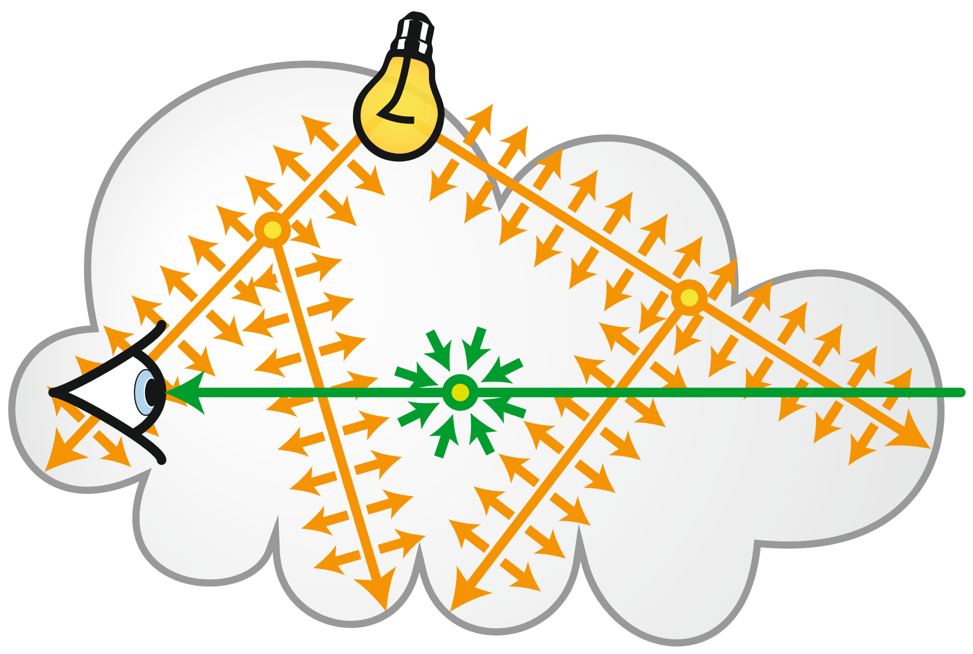
It was inspired by the spherical lights; we basically take each ray light and inflate it into a beam. This removes the singularities completely.

Spreading the Energy

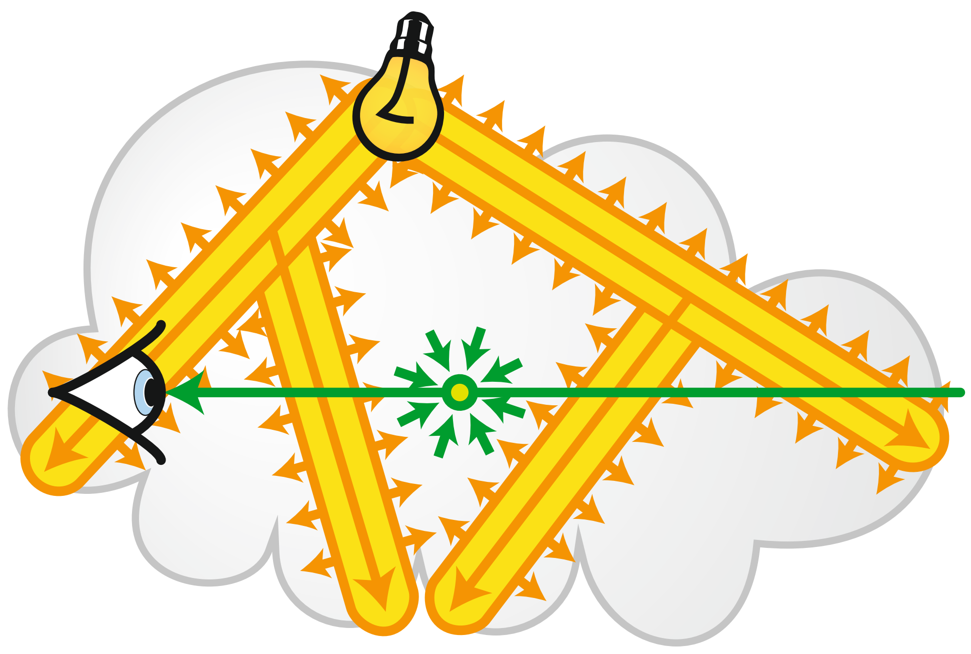
Progressive Virtual Beam Lights [Novák et al. 2012b]

- give thickness to VRLs
- use the concept of VSLs to “inflate” points on the ray

Virtual Ray Lights



Virtual Beam Lights



- remove singularities completely
- similar to the concept of VSLs

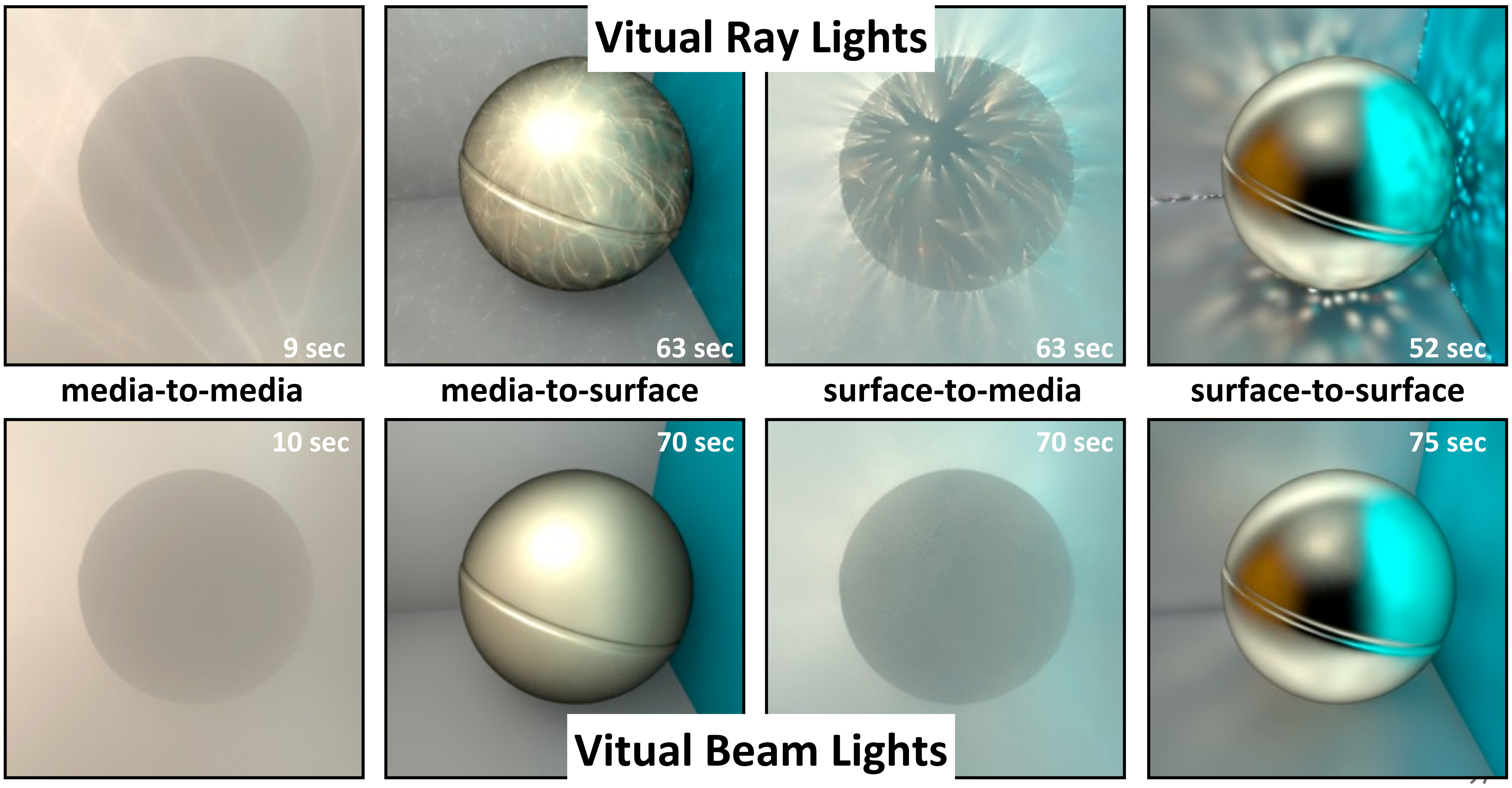
The last technique that I will mention is a follow up on the ray lights.

It was inspired by the spherical lights; we basically take each ray light and inflate it into a beam. This removes the singularities completely.

Spreading the Energy

Progressive Virtual Beam Lights [Novák et al. 2012b]

- give thickness to VRLs
- use the concept of VSLs to “inflate” points on the ray



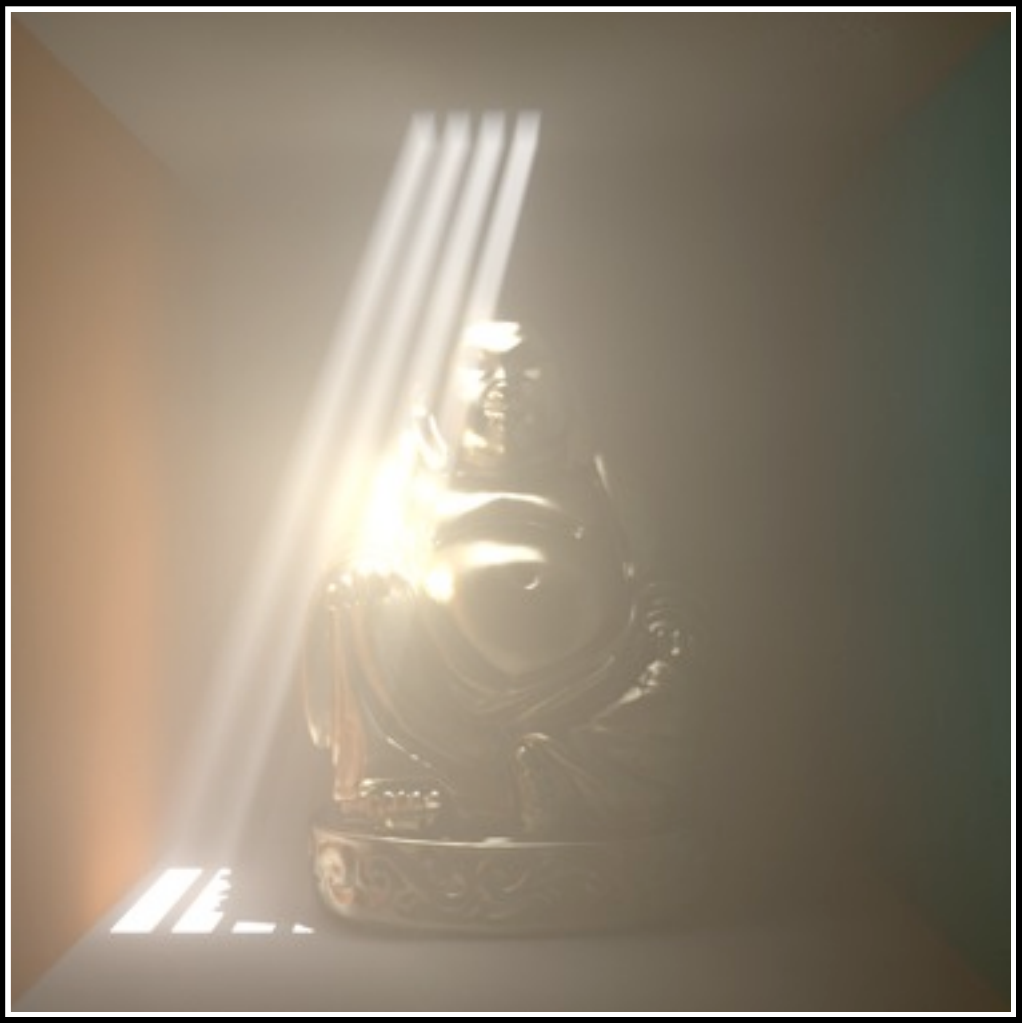
Here we have few example renderings.

I will not explain the details since we do not have time for it but on top you see some renderings with ray lights after short rendering time and you can see some artifacts.

At the bottom we have the beam lights with the same number of lighting primitives and you can see that the artifacts are not present anymore and the rendering time increases only slightly.

Spreading the Energy

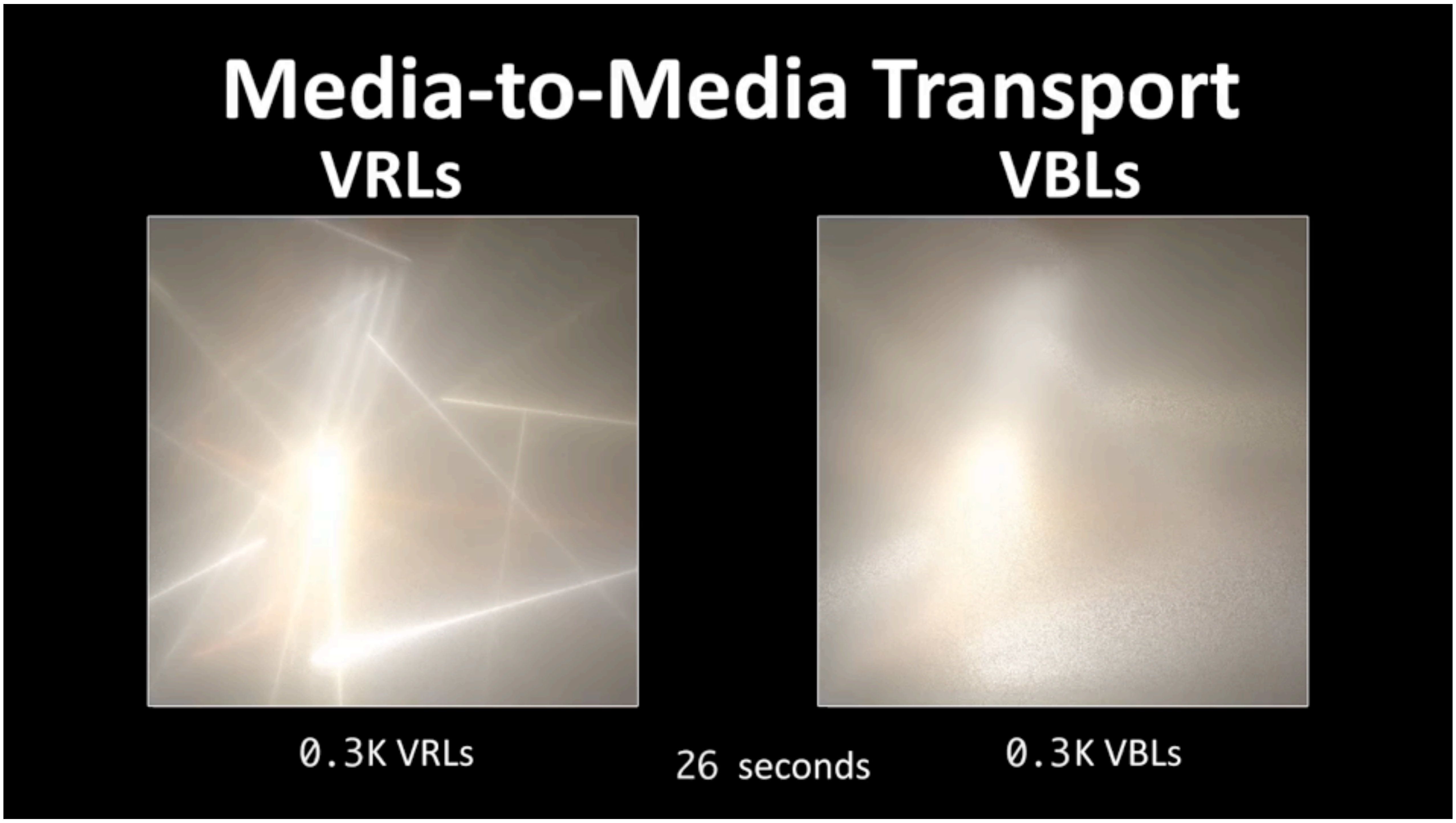
Progressive Virtual Beam Lights [Novák et al. 2012b]



Buddha Scene
homogeneous
anisotropic (HG g= 0.7)

Spreading the Energy

Progressive Virtual Beam Lights [Novák et al. 2012b]



You see a progressive rendering with the ray lights on the left and beam lights on the right.

And you can see that in each of the different light transports the artifacts are reduced and beam lights provide quality previews already after short amount of time.

Spreading the Energy

Progressive Virtual Beam Lights [Novák et al. 2012b]

All-in-all the progressive virtual beam lights combine several techniques.

We use the same reasoning to inflate the ray lights into beams, as was used for inflating point lights into spheres.

And we also formulate the algorithm progressively so that the thickness of the beams is reduced in each frame yielding convergent results.

The disadvantage is that the integration becomes again a bit more expensive. Also, some of the sharp features may be over-blurred at the beginning of the progressive rendering.

Spreading the Energy

Progressive Virtual Beam Lights [Novák et al. 2012b]

Advantages:

- energy is preserved, distributed over the volume of a beam
- no singularities
- progressively reduces the beam width -> converges to ground truth

All-in-all the progressive virtual beam lights combine several techniques.

We use the same reasoning to inflate the ray lights into beams, as was used for inflating point lights into spheres.

And we also formulate the algorithm progressively so that the thickness of the beams is reduced in each frame yielding convergent results.

The disadvantage is that the integration becomes again a bit more expensive. Also, some of the sharp features may be over-blurred at the beginning of the progressive rendering.

Spreading the Energy

Progressive Virtual Beam Lights [Novák et al. 2012b]

Advantages:

- energy is preserved, distributed over the volume of a beam
- no singularities
- progressively reduces the beam width -> converges to ground truth

Disadvantages:

- requires integration along both rays and over the solid angle
- may over-blur sharp illumination features at the beginning

All-in-all the progressive virtual beam lights combine several techniques.

We use the same reasoning to inflate the ray lights into beams, as was used for inflating point lights into spheres.

And we also formulate the algorithm progressively so that the thickness of the beams is reduced in each frame yielding convergent results.

The disadvantage is that the integration becomes again a bit more expensive. Also, some of the sharp features may be over-blurred at the beginning of the progressive rendering.

Lighting with VPLs

Ok, so we covered quite a few techniques that address high quality lighting with VPLs.

Here we have a very short summary for algorithms that handle surface illumination. All of them trade speed for quality and considering all the aspects there is no clear winner.

If you aim at speed, you do not want to just clamp, than you can use the screen space bias compensation.

For higher quality you should look into VSLs or local lights.

For unbiased rendering you can use the original bias compensation technique by Kollig and Keller, but then it might be worth considering a completely different algorithm.

Lighting with VPLs

Comparison of techniques handling surfaces only

	Speed	Quality	Implementation
Bounding only	✓	✗	✓
Bias Compensation [Kollig and Keller 2004]	✗	✓	✓/✗
Local Virtual Lights [Davidovič et al. 2010]	✓/✗	✓/✗	✓/✗
Screen-space Bias Comp. [Novák et al. 2012]	✓	✓/✗	✓/✗
Virtual Spherical Lights [Hašan et al. 2009]	✓/✗	✓/✗	✓/✗

✓ good, easy
✗ bad, difficult

Ok, so we covered quite a few techniques that address high quality lighting with VPLs.

Here we have a very short summary for algorithms that handle surface illumination. All of them trade speed for quality and considering all the aspects there is no clear winner.

If you aim at speed, you do not want to just clamp, than you can use the screen space bias compensation.

For higher quality you should look into VSLs or local lights.

For unbiased rendering you can use the original bias compensation technique by Kollig and Keller, but then it might be worth considering a completely different algorithm.

Lighting with VPLs

Comparison of techniques handling participating media

Here we have an analogous comparison of techniques handling the media.

Here the situation is a bit easier, both the ray and beam lights seem to perform better than algorithms using point lights.

This is simply because in media we can distribute the energy along lines and this greatly reduces the problems.

Lighting with VPLs

Comparison of techniques handling participating media

	Speed	Quality	Implementation
Bounding only	✓	✗	✓
Bias Compensation [Raab et al. 2008]	✗	✓	✓/✗
Approximate Bias Comp. [Engelhardt et al. 2012]	✓	✓/✗	✗
Virtual Ray Lights [Novák et al. 2012]	✓/✗	✓/✗	✓/✗
Virtual Beam Lights [Novák et al. 2012]	✓/✗	✓	✓/✗

✓	good, easy
✗	bad, difficult

Here we have an analogous comparison of techniques handling the media.

Here the situation is a bit easier, both the ray and beam lights seem to perform better than algorithms using point lights.

This is simply because in media we can distribute the energy along lines and this greatly reduces the problems.

References

- [**Davidovič et al. 2010**] - *Combining Global and Local Virtual Lights for Detailed Glossy Illumination*, ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia '11), 2011
- [**Engelhardt et al. 2012**] - *Approximate Bias Compensation for Rendering Scenes with Heterogeneous Participating Media*, Computer Graphics Forum (Proceedings of Pacific Graphics '12), 2012
- [**Georgiev and Slusallek 2010**] - *Simple and Robust Iterative Importance Sampling of Virtual Point Lights*, Eurographics '10 (short papers), 2010
- [**Hašan et al. 2009**] - *Virtual Spherical Lights for Many-Light Rendering of Glossy Scenes*, ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia '09), 2009
- [**Kollig and Keller 2004**] - *Illumination in the Presence of Weak Singularities*, Monte Carlo and Quasi-Monte Carlo Methods, 2004
- [**Novák et al. 2011**] - *Screen-Space Bias Compensation for Interactive High-Quality Global Illumination with Virtual Point Lights*, I3D'11 Symposium on Interactive 3D Graphics and Games, 2011
- [**Novák et al. 2012a**] - *Virtual Ray Lights for Rendering Scenes with Participating Media*, ACM Transactions on Graphics (Proceedings of SIGGRAPH '12), 2012
- [**Novák et al. 2012b**] - *Progressive Virtual Beam Lights*, Computer Graphics Forum (Proceedings of EGSR '12), 2012
- [**Raab et al. 2008**] - *Unbiased Global Illumination with Participating Media*, Monte Carlo and Quasi-Monte Carlo Methods, 2006
- [**Segovia et al. 2006**] - *Bidirectional Instant Radiosity*, Proceedings of the 17th Eurographics Workshop on Rendering, 2006
- [**Segovia et al. 2007**] - *Metropolis Instant Radiosity*, Proceedings of Eurographics '07, 2007